Making Robots With The Arduino - Part 1

SERVO MAGAZINE

FOR THE ROBOT INNOVATOR

www.servomagazine.com

November 2010

Dancing Bots

So, You Think Your Bot Can Boogie?

GoPHR

Homebuilt Prototype Platform

Inexpensive Expandable

HoverBot Wrap-up

You Picked A Fine Time To Leave Me Loose Wheel!

U.S. $5.50 CANADA $7.00

11

0 71486 02422 4


KECON
Let your geek shine.
Meet Peter Madsen and Kristian von Bengtson, two of the brains behind the Copenhagen Suborbitals project. Peter and Kristian used SparkFun's Logomatic board to record vital data during the testing of their rocket. Ultimately, Kristian will man the spacecraft as it is launched into suborbital space.
Whether you're outfitting a shirt with LEDs, or sending a rocket into orbit, the tools are out there. Explore a new world and let your geek shine too.
sparkfun ELECTRONICS
Sharing Ingenuity
WWW.SPARKFUN.COM
© 2010 SparkFun Electronics, Inc. All rights reserved. All other trademarks contained herein are the property of their respective owners. Follow Peter and Kristian as they near their goal of launching a manned rocket into space at www.copenhagensuborbitals.com. Good luck and have a safe flight!

Power Your Connected Graphics Solution With the PIC32 Microcontroller

Microcontrollers
Digital Signal Controllers
Analog
Memory

Connectivity and graphical user interfaces are essential in today's applications. You're challenged to deliver intuitive, high impact, connected solutions while maintaining flexibility to support several different product options. Microchip's PIC32 series of 32-bit microcontrollers offer the right performance, memory size and peripherals to help achieve your goals.

With 1.56 DMIPS/MHz performance topping any device in its category, up to 512 Kbytes of Flash, 128 Kbytes of RAM and integrated connectivity peripherals like Ethernet, CAN and USB the PIC32 can deliver the mix of performance and flexibility needed to help you meet your design challenges.

Microchip gets you there with:

- PIC32 Starter Kits – Standalone easy to use development boards with integrated debugger/programmer
- Multimedia Expansion Board – The most complete user interface development solution in it's class – enabling development of highly interactive, graphics and audio-based interfaces with WiFi connectivity – a modular add-on to any PIC32 Starter Kit.
- Microchip's FREE Graphics and Connectivity libraries and code examples – Eases your development effort and speeds your time to market

GET STARTED IN 3 EASY STEPS

1. Purchase PIC32 Ethernet Starter Kit and Multimedia Expansion Board
2. Download MPLAB® IDE
3. Start designing!

www.microchip.com/graphics

PIC32 Ethernet Starter Kit – DM320004

Multimedia Expansion Board - DM320005

www.microchip.com/graphics

MICROCHIP

worldmags

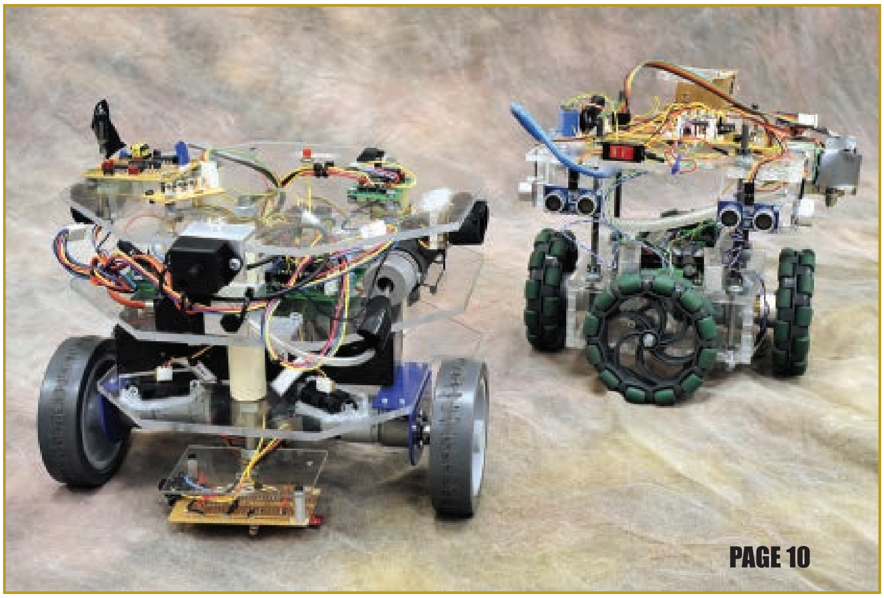
SERVO MAGAZINE

11.2010
VOL. 8 NO. 11

Did you know that each article in *SERVO Magazine* has its own webpage? It's where you go for downloads, comments, updates, corrections, or to link to the article in the digital issue. The unique link for each webpage is included with the article.

You can also visit article pages from back issues at **www.servomagazine.com**. Just select the ***Past Issues*** tab from the ***About SERVO*** drop down menu, click the Table of Contents link, and the article name.

Columns



PAGE 10

The Combat Zone...

Features

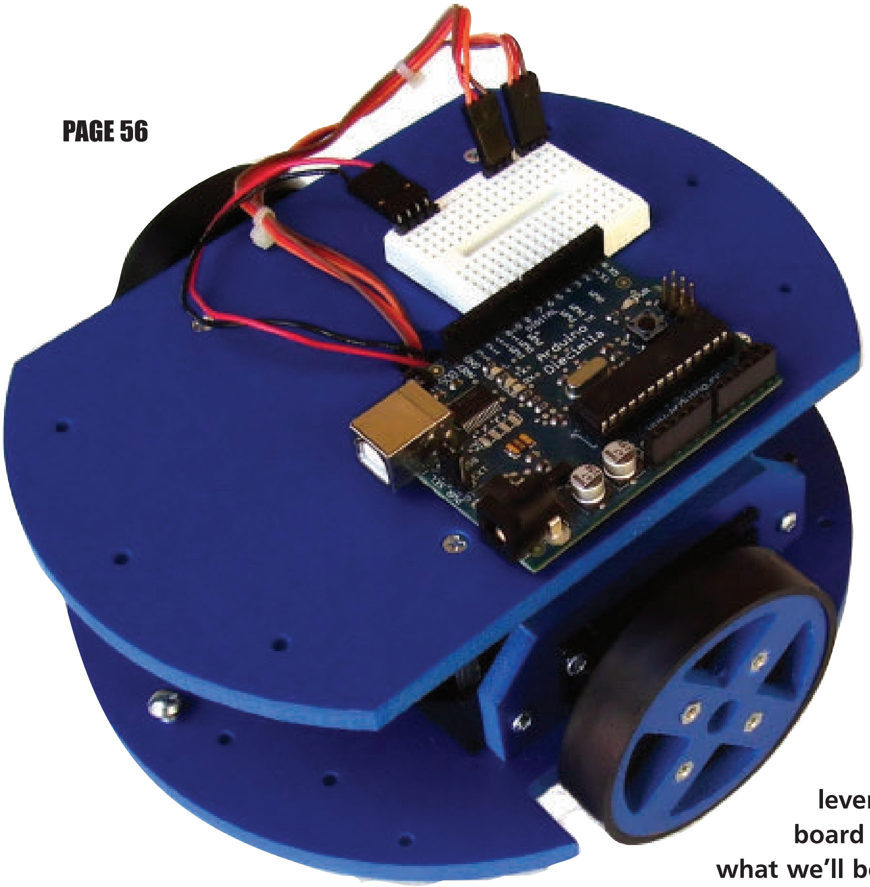
Events

Departments

SERVO Magazine **(ISSN 1546-0592/CDN Pub Agree#40702530)** is published monthly for $24.95 per year by T & L Publications, Inc., 430 Princeland Court, Corona, CA 92879. PERIODICALS POSTAGE PAID AT CORONA, CA AND AT ADDITIONAL ENTRY MAILING OFFICES. POSTMASTER: Send address changes to **SERVO Magazine, P.O. Box 15277, North Hollywood, CA 91615** or Station A, P.O. Box 54, Windsor ON N9A 6J5; cpcreturns@servomagazine.com

In This Issue ...

PAGE 56



PAGE 36

Mind / Iron

by Bryan Bergeron, Editor

Practical Service Robotics

If you're a robotics enthusiast, you've probably at least considered picking up one of the robot vacuum cleaners. I have. But I haven't purchased one because I haven't found a robot vacuum cleaner – the most popular form of service robot on the planet – that can replace a handheld vac. I also suspect that many consumers share my sentiments, as well.

My requirements for a home robot vacuum cleaner are simple enough – the vac simply has to work like the litter cleanup robots on Star Wars. If you recall, the small, black robots scurried about in search of litter. When a robot spotted something in the hallway, it would suck it up and then scurry away in search of the next object. Sounds simple enough, doesn't it?

I want a robot that will recognize that I dropped a bowl of popcorn on the carpet, head to the sight of the disaster, suck up the popcorn, and then return to the charging station. As a bonus, if the vac happened to scoop up a coin, transistor, or earring, I'd like the valuable stored in a separate bin in the vacuum's body.

However, in my recent survey of what's available on the market, nothing came close to my ideal. The old standby — the iRobot Roomba — can swirl around a room for an hour before happening upon a pile of crumbs. The more expensive robot vacs such as the Neato XV-11 and Samsung Navibot excel at navigation – meaning they can avoid obstacles and almost guarantee full floor coverage. However, even the $650 Navibot can't spot a fresh spill, head directly to the area, clean it up, and then move out of sight. No, for quick spills, robot vacs still can't hold their own against a $35 handheld vac.

So, what's the holdup? Why can't one of the big robotics manufacturers come up with software and a hardware platform that can effectively replicate a human-directed hand vac? Think about how you'd go about devising sensors and the control program for such a vac. One option would be to scan the rooms with ceiling-mounted cameras. You'd have to program the system to ignore people, pets, and toys. There's also the issue of changes in lighting with time of day. Another potential problem is how to recognize – and ignore – spilled liquids. You don't want your robot vac making a mess by rolling in spilled chocolate milk or red wine.

You might also consider a sensor that's specific for organic material. Perhaps a sensor that's sensitive to methane – produced by rotting organic material – could help a robot vac hone in on day old crumbs.

In the Star Wars movie, the dog-like litter robot worked on shiny, highly waxed floors. Perhaps a similar environment would make the work of a modern robot vac easier. Consider a laser radar that's fired just above and parallel to the floor. Anything between the robot and floorboard that's only a few millimeters in height or width is suspect and requires closer inspection.

Unfortunately, my hypothetical solutions to making a robot vac more useful aren't necessarily economically feasible. What would you do to bridge the functionality gap? What are your must-haves for a robotic vac? Share your list and, if you have them, photos of your creation with *SERVO* readers. **SV**

FOR THE ROBOT INNOVATOR **SERVO** MAGAZINE

Published Monthly By
T & L Publications, Inc.
430 Princeland Ct., Corona, CA 92879-1300
(951) 371-8497
FAX **(951) 371-3052**
Webstore Only **1-800-783-4624**
www.servomagazine.com

Subscriptions
Toll Free **1-877-525-2539**
Outside US **1-818-487-4545**
P.O. Box 15277, N. Hollywood, CA 91615

PUBLISHER
Larry Lemieux
publisher@servomagazine.com

ASSOCIATE PUBLISHER/ VP OF SALES/MARKETING
Robin Lemieux
display@servomagazine.com

EDITOR
Bryan Bergeron
techedit-servo@yahoo.com

CONTRIBUTING EDITORS
Jeff Eckert, Jenn Eckert, Tom Carroll, David Geer, Dennis Clark, R. Steven Rainwater, Fred Eady, Kevin Berry, Greg Intermaggio, Paul Verhage, Bryce Woolley, Evan Woolley, Alan Federman, Pete Smith, Gordon McComb

CIRCULATION DIRECTOR
Tracy Kerley
subscribe@servomagazine.com

MARKETING COORDINATOR WEBSTORE
Brian Kirkpatrick
sales@servomagazine.com

WEB CONTENT
Michael Kaudze
website@servomagazine.com

ADMINISTRATIVE ASSISTANT
Debbie Stauffacher

PRODUCTION/GRAPHICS
Shannon Christensen

X-TREME geek

x-treme Geek is wireD in to what you want.

You want that Big Reaction when your loved one opens their gift. We guarantee you'll get it when you shop with us. From never before seen modern marvels to kitschy cool blasts from the past, X-treme Geek has the gifts to make this holiday the best one yet. Shop x-tremegeek.com.

2500941.........$16.95
EchoBot Voice Messenger

- Detects movement up to 3 feet away

Computer not included

EchoBot Delivers a Voice Message when Somebody Moves Nearby

Record messages for your friends and coworkers, and the EchoBot will play them back when they approach. Up to ten seconds of audio will play back when someone triggers the motion detector (built into the "eye," of course). Bendable legs and suction cup feet make it easy to position the EchoBot anywhere around your home, office or car. Colors vary. Size: 7" tall.

A Carpentry Kit for a Robot?

NEW

Maybe early robots were made of wood....or maybe it's just an awesome idea who's time is long overdue. This kit is supplied with pre-punched wooden boards, gears, shafts, switch, motor, battery holder, and all necessary parts to build your own wooden robot, complete with blinking eyes. Includes easy-to-follow instructions. Requires screwdriver and long nose pliers for construction, plus two AA batteries - all not included. Recommended for ages 10 and up.

3152147.....$19.95
Robomech Wooden Kit

⚠ WARNING:
CHOKING HAZARD—Small Parts.
Not for children under 3 yrs.

POW Robot T-Shirt

NEW

From R2-D2 and C3-PO to Johnny 5 and Rosie, there are few things cooler than robots. Now you can proudly sport your robot fandom in this cotton tee. Accented by comic book style POW background, the robot pictured on this shirt will WOW your friends while you look cool, calm, and comfortable. Made in America and printed with environmentally friendly inks. Color: Heather

POW Robot | **$24.95**

3200150........Medium
3200151............Large
3200152........X-Large
3200153......XX-Large

Virtual Guard Dog Protects Your Home 24/7

Dog not included

- Adjustable volume & sensitivity
- 5¼"L x 5¾"W x 7½"H; 6' cord

Switch from barking to tranquil rainforest sounds for a soothing audio backdrop for a party!

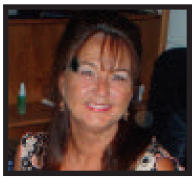
1320672.........$89.95
Rex Plus, The Electronic Watchdog

Few security systems are as intimidating as an angry barking dog. The Rex Plus security system "sees" through walls and doors to detect when someone approaches your home and then it "barks" just like a real German Shepherd. The startlingly realistic barking increases in intensity as the interloper gets closer. You get inexpensive, reliable, 24-hour protection that's perfect for extended absences.

3200103.....$89.95
The Wizard's Wand Universal Remote

Cast A Magic Spell Over Your Electronics!

To truly rule the room and couch kingdom you must have a wizard on your side...or better yet, be a wizard yourself. With this vibrating wand as universal remote, controlling your home entertainment systems is simple sorcery. A flick of the wrist and a spin of your magical wand changes the channel, volume, track, and more, including rewind and fast forward. A total of 13 programmable commands are controlled by circular movements, up and down gestures, or back and forth whisps of this vibrating wand. It looks like magic, but it's really technology — the same accelerometer technology used in Wii remotes. With practice you'll master a level of wizard like skill to impress all your friends. The remote is recognized by almost every piece of modern home entertainment apparatus. Size: 39cm x 6.5cm x 3.8cm. Weight: 295g.



Robytes

by Jeff and Jenn Eckert

Oil Sucker Prototype

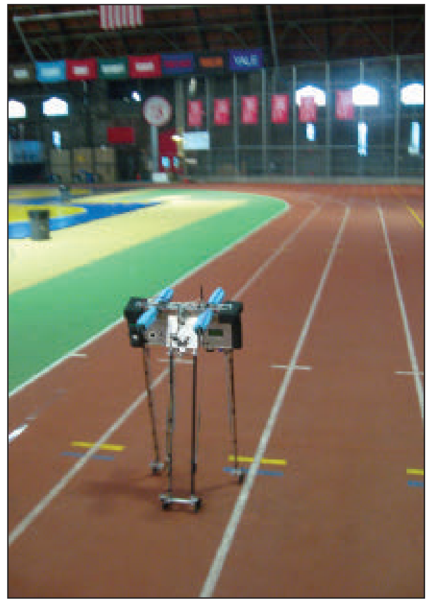
The Seaswarm bot team autonomously cleans up oil spills.

A few weeks ago, the folks at MIT's Senseable City Lab (**senseable.mit.edu**) tested a prototype robotic oil skimmer dubbed Seaswarm in the Charles River. You might not think of the Charles as a hotbed of oil spills, but it happens now and then. In fact, the Cambridge-based Mirant Kendall power plant has reported five separate oil-related incidents since 1999 and has been cited for 35 effluent violations since 2006. We're not talking about Gulf of Mexico-scale spills, but it's got to be pretty annoying to the locals. In any event, Seaswarm offers a potential solution for any level of environmental mishap. The concept is based on releasing a swarm of skimmers that can work autonomously and continuously for weeks. In operation, each bot employs a rolling conveyor belt of thin nanowire mesh to absorb the oil, squeezes the oil into a collection chamber of some sort, and "digests" it on site. The swarm communicates and coordinates their operations via GPS and WiFi, thereby establishing an organized system. The exact digestion process is unclear at this point, and the prototype doesn't appear to include that capability. However, related literature mentions compressing the belt to remove the oil and subsequently burning it, so perhaps that's the next evolutionary step. In any event, Seaswarm measures 16 x 7 ft (4.9 x 2.1 m), uses two square meters of solar panels for propulsion, and draws only 100W of current. According to MIT, 5,000 of the relatively inexpensive Seaswarms working for a month would be capable of cleaning up the Deepwater Horizon spill. Prototype updates are in the works, so stay tuned.

Bot Sets Walking Record

It's official. The Ranger robot — developed at Cornell University (**www.cornell.edu**) — has bumped Boston Dynamics' BigDog from the top spot for nonstop walking by an untethered, legged robot. BigDog previously set the record with 12.8 miles (20.6 km), but Ranger took the championship by taking 65,185 steps on an indoor track at Barton Hall, covering 14.3 miles (23 km) on a single charge over a period of a little over 10 hours, 40 minutes. For good measure, it used only about a penny's worth of electricity per three miles of travel. Given that Ranger isn't designed to do much else, there's not a lot more to say about it. It's steered with a model airplane remote control, employs six on-board microprocessors, and travels at about 1.34 mph (2.15 km/h) using a 25.9V lithium-ion battery.

Ranger strolls to a world record on an indoor track.



Bot Trains Chopper Pilots

The first few hours of a helicopter pilot's training are known to be particularly intense and dangerous — more so than with fixed-wing aircraft. This is true for a variety of reasons, among which are the complexity of the machinery, the environment in which they are flown, and the fact that a student will typically learn his stuff inside a Robinson R22 (which unfortunately wasn't designed to be a trainer). However, the folks at the Max Planck Institute for Biological Cybernetics (**www.kyb.mpg.de**) have teamed up with KUKA Roboter (**www.kuka-robotics.com/germany/de**) and Heli Aviation (**www.heli-aviation.de**) to make chopper training both cheaper and safer. The result is Heli Trainer, introduced at the

The Heli Trainer helicopter flight training robot.

2010 Berlin Air Show. The robotic trainer is based on the KUKA robot type KR 500 TÜV, which was further developed for motion simulation by the Planck Institute. Attached to the six-axis, heavy-duty bot is a Buimbal Cabri G2 helicopter cell which has space for two people who can learn realistic helicopter maneuvers in an "original" cockpit. According to a Planck rep, "With the Heli Trainer, a pilot trainee requires less time to develop a feel for movements, understands the consequences of his flight control actions better, and learns maneuvers in a safe environment with a steeper learning curve." It also looks like a lot of fun.

Bot Bound for Tranquility

3Model of the Astrobotic Red Rover lunar bot.

It's been more than 40 years since a human being walked in the lunar Sea of Tranquility, and it's likely to stay that way for some time. However, Carnegie Mellon spinoff Astrobotic Technology (**www.astrobotictech.com**) is scheduled to drop in a rover in April 2013 to revisit the landing site vicariously. In addition to the thrill of the challenge, the company has plenty of incentive: NASA has offered to pay up to $10 million for data collected during a commercial lunar mission, and Google is offering $20 million as its Lunar X prize for the first team to land a robot on the moon that (a) travels more than 500 m (1,640 ft) and (b) sends back high-def images and video. There is also an additional $5 million up for grabs if the bot performs some other tricks such as traveling more than 5,000 m (3 mi), detecting ice in a crater, surviving a lunar night, and so forth. In addition, the State of Florida is kicking in $2 million for the privilege of providing the launch site, and Caterpillar is providing both financial and technical support. The "Tranquility Trek" mission will star Astrobotic's Red Rover vehicle which weighs in at 160 lb (72.6 kg) and stands about 5 ft (1.5 m) tall. Part 1 of the mission will last 10 to 12 days, until sunset cuts off the rover's power. It will then hibernate as temperatures drop to nearly -300°F (-149°C). If it survives it will wake up at sunrise which is about two weeks later and continue its mission. For details and vids, visit **astrobotic.net/activities**.

Robotic Publisher Changing Industry

The Espresso Book Machine® from On Demand Books, LLC, paired with a Xerox copier/printer.

If you enjoy dropping into your local Barnes & Noble, sipping some overpriced coffee, and nibbling on a macadamia nut/chocolate chip cookie while you leisurely browse through prospective book purchases, you may be disappointed as such behemoth bookstores are gradually being replaced by little robotic book huts with — if you're lucky — coffee and snack machines. A robotic book factory — ironically called the Espresso Book Machine — has been unleashed by On Demand Books (**www.ondemandbooks.com**) and is functioning in "dozens of locations worldwide" with more locations coming soon. Basically what we're talking about is a machine that automatically prints, binds, and trims a book at the point of sale, generating perfect-bound, library-quality books within a few minutes. For example, a 300 page book can be spit out in about four minutes with a consumables cost of about a penny per page. It can handle books in the range of 40 to 830 pages using letter-size or tabloid coverstock, toner, ink, and glue. The covers are full color and said to be indistinguishable from books produced by traditional processes. In addition, using the "EspressNet" network, each machine can access more than a million copyrighted titles from some 6,500 publishers, and three million more public domain titles via Google Books and other sources. The main obstacle seems to be the price which will run you just under $100,000, which doesn't even include the compatible Xerox 4112 or Kyocera FS-9530DN printers. Still, it ain't cheap to maintain the average B&N which sits on 25,000 square feet and carries up to 200,000 titles, so economic considerations may make the device practical for educational institutions and libraries, small bookstores, and other locations. SV

GEER HEAD

by David Geer

Contact the author at geercom@windstream.net

University of Akron Competition Robot Roundup

Events such as RoboGames attract bright minds and hungry competitors from across the country. The University of Akron's student roboticists were well prepared with well-designed, efficient, and intelligent mobile robots.

PiRATE

University of Akron roboticists designed and created PiRATE with a focus on competing in the NASA sponsored Lunabotics Mining competition. In this competition, robots maneuver to a given locale, collect moon soil, and return to deposit it in a pre-appointed collection container or bin. The activity is a staged simulation of the actual Lunar Rover collecting Lunar regolith on the surface of the moon.

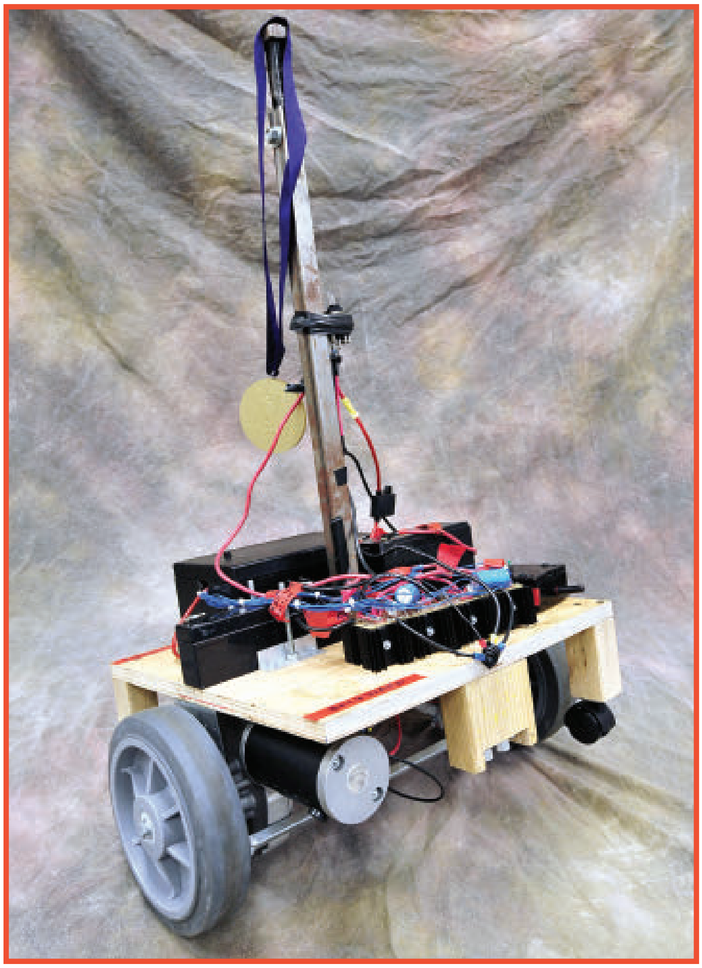
A differential drive moves the steel robot on tracks like they would on a tank. "The tracks were made from table chain which are conveyor belts used for transporting items in manufacturing systems," says Dr. Tom Hartley, robotics advisor, University of Akron. Linear actuators raise and lower the front facing plow used to collect the moon soil.

The robot's operators control it remotely via Wi-Fi. The operators view the robot from a separate area via cameras mounted in the arena. Three wired Ethernet-based cameras enable the robot's own sight. The roboticists mounted them on the front and back of the robot, and adjacent to the plow at the proper angle so its operators could view the contents.

The builders created the operator interface in Visual Studio. The robot employed PIC and Arduino

Tom Hartley, professor of computer and electrical engineering at the University of Akron, points out the simple and easy-to-implement design of Lunabot PiRATE (Piloted Remote All-Terrain Excavator) — a lunar mining robot. PiRATE's bucket, for instance, can scoop up 15 kg of lunar sand by remote control via Internet from miles away. Lunabot's lightweight, compact lithium battery is markedly more efficient than its conventional lead counterpart and its lightweight polyurethane tracks efficiently maneuver across lunar surfaces.

Lunabot PiRATE, with engineering student designers (l-r): Tom Vo, Richard Johnson, and Bruce Haas.



This remote controlled, two-wheel, self-balancing robot maneuvers with ease at a speed of about 5 mph. UA's BalanceBot took first place at this year's RoboGames held in April in the Bay Area.

microcontrollers which were programmed using the C language. The Arduino addressed the Wi-Fi communications and robot commands. It would pass messages useful to the PIC controller on to it so it could control the motor driver and linear actuator system. Operators use a GUI on a PC to control the robot's speed and send motion related commands to the tracks and the plow. Interestingly, the final version of the robot is able to turn itself off if the operators lose communication with it.

BalanceBot

The Akron students targeted the BalanceBot competition at RoboGames 2010. In the competition, RoboGames requires the operators to drive the two-wheeled, self-balancing robot — which is like the Segway in many respects — via remote down a straight course. They must then stop the robot and have it balance itself for a while. Then, the roboticists must return it to the course's starting point.

The wooden platform robot houses most of its electronics inside. The motors that drive the robot's eight inch wheels are of the permanent magnet geared variety. The robot's control system is quite sophisticated, as college competition robots go. The bot's ultrasonic transducers, (which are affixed to its bottom) provide sensory feedback on the platform's angular position.

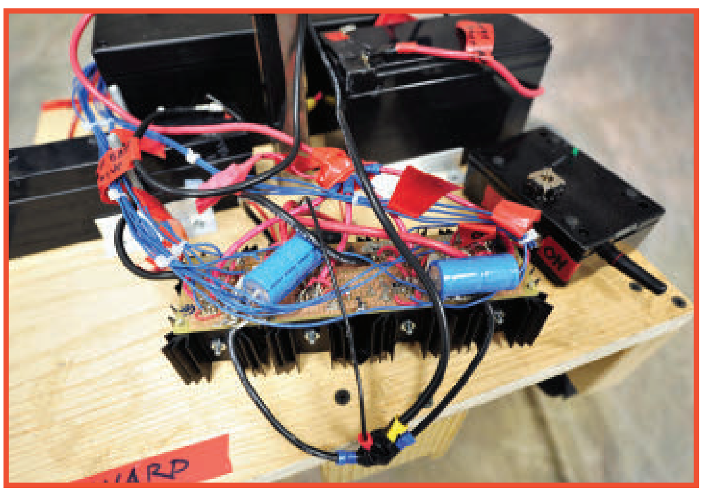
A dedicated dsPIC33FJ64MC804 microcontroller computes and transmits the transducer sensory data to the digital controller to balance the robot. The students programmed the PIC controller in C and designed the digital control system by modeling the robot's dynamics.

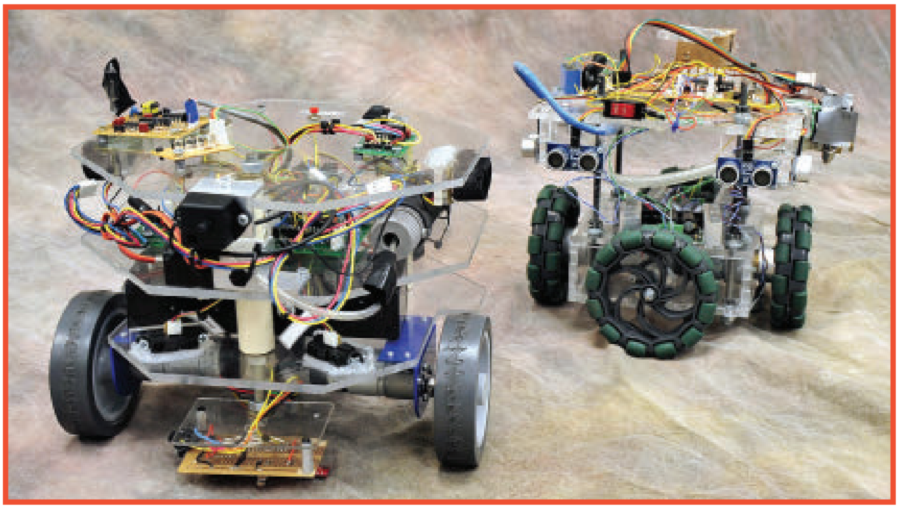
Dr. Hartley drills down to the control systems finer points:

> "This system used a modified PID controller containing two poles and two zeros and implemented as a difference equation. The controller also incorporated the velocity command signal from the wireless transceiver to drive the robot from one position to another.
>
> The students implemented the digital controller using a separate dsPIC33FJ64MC804 that they had programmed in C. The custom designed H-bridge motor driver received the commands from the controller and commanded the motor torques through the H-bridge. The students controlled the H-bridge driver using a dedicated dsPIC33FJ64MC804.
>
> A separate microcontroller received the wireless signal velocity commands. The roboticists connected all the microcontrollers together with a dedicated I²C network. They used three dsPICs due to the necessary high speed of the math-intensive control loop."

Operators communicated with the robot via wireless

Close-up of BalanceBot components.





UA student-designed fire fighting robots — two-wheel Jake (left) and four-wheel Rock Lobster (right) both incorporate UV Tron sensors that can find a fire by sensing its ultraviolet photons. Rock Lobster won the Trinity College Fire Fighting competition in 2010, and Jake placed second at the 2010 RoboGames.

University of Akron engineering alumnus Joe Davis with fire fighting robot "Jake." This two-wheeled robot can autonomously locate and extinguish an indoor fire.

using a 433 MHz transceiver. Using the current design, the operators use the remote control to send four eight-byte data packets every 250 ms to the robot. Bytes 1-4 specify the speed for directional movement.

"Byte 1 specifies forward movement, byte 2 is for backward motion, byte 3 is for left movement, and byte 4 is for right movement. Byte 5 authenticates the joystick. Bytes 6 and 7 enable cyclic redundancy checking code," explained Dr. Hartley.

Jake

Jake — fire fighting robot — was implemented with the capacity to search a mock house for a candle and put the candle out without human assistance. This competition was also part of RoboGames 2010. A goal of the competition included putting out the candle in the shortest time possible.

"The competition itself is based on the premise that robots could be used to patrol houses and increase fire safety," says Dr. Hartley. Jake won second place at RoboGames.

The students cut the polycarbonate robot out using a laser cutter. The robot's many sensors included ultrasonic, long-range Sharp IR sensors, and short-range Sharp IR sensors in conjunction with a program that mapped the house to determine the robot's current location.

"The robot moved from points in the house, always keeping track of its position on an internal map. The robot employed a UV-tron sensor to detect the flame of the candle and then thermopiles to find the candle's exact location," commented Dr. Hartley.

Using a microcontroller-actuated VersaValve to turn the CO_2 valve, the robot could put out the fire and turn the CO_2 back off again. Dr. Hartley discussed this robot's controller intricacies:

> "The robot used a PIC24FJ32GA002-E/SP which is a 16-bit processor for onboard control. The PIC communications with the Devantech motor controller used an I^2C bus. The EMG30 DC gear motors have a one degree resolution optical encoder which was also used for position command feedback when the robot was in motion."

The University of Akron also produced Rock Lobster — another fire fighting robot — tasked with putting out a candle in a mock house. This robot used omni-directional wheels, limiting the robot's need to turn and greatly increasing its speed. This robot also used CO_2 to snuff out the candle's flame.

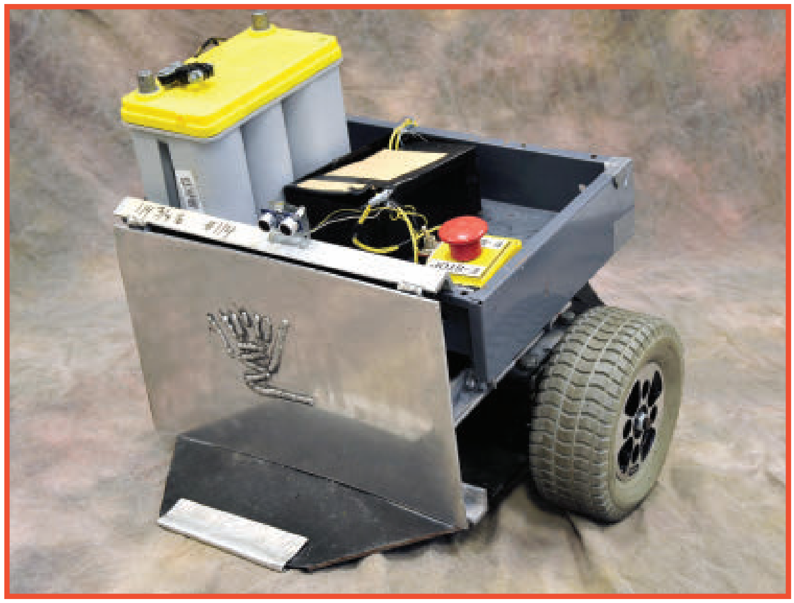
Resource

University of Akron, College of Engineering
www.uakron.edu/engineering

CombatBot and the Titanium Super Heavyweight 340 lb Combat Robot

CombatBot entered the RoboGame's 60 lb battle

This super heavyweight combat robot features a titanium shield and weighs in at 340 lbs. Two of its student designers are (l-r) Richard Johnson and Bruce Haas.



CombatBot's simple looks are deceiving. The 120 lb Sumo robot can autonomously find and push away an opponent. Unpretentious in appearance, this robot easily won the heavyweight Sumo competition at the National Robotics Challenge in 2010.

division. The operators control the robot and its weapons wirelessly. The students designed the robot as a mobile wedge that can ram underneath other robots to flip them over.

The titanium super heavyweight combat robot was a three foot by four foot by one foot titanium box. "The titanium was grade 5 titanium donated by the Defense Metals Technology Center," says Dr. Hartley. The weapons included a pneumatically actuated ram with a hardened point. DeWalt drill motors drove the robot.

An Atmega328p microcontroller observed changes in the incoming signals from a 2.4 GHz receiver and turned on the super heavyweight robot's motor driver relays or the pneumatic valves.

Conclusion

Clearly, the University of Akron has an active robotics program, keeping young engineers striving for knowledge as they solve some difficult robotics problems and have fun at the same time. **SV**

Photos by Scott Horstman for The University of Akron. Captions courtesy of the University of Akron.

Our resident expert on all things robotic is merely an email away.
roboto@servomagazine.com

Tap into the sum of *all human knowledge* and get your questions answered here! From software algorithms to material selection, Mr. Roboto strives to meet you where you are — and what more would you expect from a complex service droid?

ASK MR. ROBOTO

by
Dennis Clark

Christmas is coming and I have a list of robotic toys to choose from again this year which included parts and boards from SparkFun, motors from Pololu, motherboards, and all kinds of other cool stuff. Many of you readers know that I have been a fan of the ATmega line of processors for small robots for a while now. I like the Eclipse IDE, and the free and open source gcc compilers and AVR libraries. Then, I discovered the Arduino environment and that was very cool too. What many of you don't know is that I have been a PIC fan for many years. Microchip has a huge number of parts and many very good third-party compiler companies for C, Basic, and whatever. I've gone over to the AVR line because those parts are supported on my favorite Platform Mac OS X. I have kept an old laptop hidden in the mad scientist's lab running Windows 2000 and my MPLAB setup to program my PICs. Lately, I've been running MPLAB 8.5 in Parallels Desktop on my iMac. It's my secret "shame."

However, the world is about to change.

At their Masters Conference this August, Microchip demonstrated their MPLAB X IDE. This IDE is based on the Java NetBeans platform and is OS agnostic. Microchip is busy porting their PIC compilers to this new platform and they too will be OS agnostic. I've used the Beta release and it is indeed sweet. It has many new upgrades and cool things that they could not do with their older IDE. However, the BEST thing about it is that their hardware suite has had their drivers ported too. Holy PIC, Batman! We're about to be able to run a supported Microchip IDE and program our PIC24FJ64GA006 with our ICD3 on Mac OS X 10.5, Linux, and Windows! For those of you that are like me, this is the best news we've heard all year (which I admit, doesn't have much competition), but still, Microchip has heard and responded. Rumor has it that the MPLAB X will release in January 2011 and it will be free. My fingers are crossed — I'm testing the Beta version now and I can't wait for the release!

Life is good.

Q. I've gotten tired of my old fashioned way of programming my robots using a big loop and doing things one at a time. I keep missing things and messing up. I don't want to have to buy a fancy motherboard running Linux, or something even more expensive to get multitasking. Someone said I don't have to do that. Can you tell me how to get multitasking on a simple AVR ATMEGA128? That would be so cool, Mr. Roboto!

— Robo Bob

A. Robo Bob, your sources are correct! You can indeed get a form of multitasking working on your favorite small processor. The magic is called "state machines" which you may have seen written as FSM, or Finite State Machine. This type of programming is the mainstay of the embedded programming world. Every embedded programmer needs this tool in their toolbox if they are going to succeed. Indeed, every robot programmer needs it too!

When you use what is called a real time operating system (RTOS) such as Linux, uCOS, VX Works, or the like, you get what is called a preemptive multitasking operating system where you define threads that all run concurrently. There are a couple of terms here you may not be familiar with; the first is preemptive multitasking. In a nutshell, this means that the operating system has a periodic interrupt — usually in the 1 ms to 10 ms time range — that stops whatever is happening and switches to another task thread. Your code has no choice about this; it just happens. The other term is "concurrent." This is a very important concept. Frequently, you hear folks say that many threads run at the same time, or simultaneously. This is incorrect (unless you have a multi-processor system). Only one thread is running at a time, for whatever the *timeslice* period has been defined. The program gives the illusion of simultaneous execution because the tasks switch around so fast. This is called *concurrent* processing,

When you program using an FSM, your program needs to define and handle when control passes from one part of the program to another. In some ways, this could be called "cooperative" multitasking which means that the different parts of the program cooperate to avoid any one part starving another part of the program for processing time. We use state machines to handle this. I won't go into a discussion about *Mealy* vs. *Moore* state machines. We'll be using a combination of them, as well as a time modified version, that Rod Brooks and his MIT team call the *Modified* FSM, or MFSM for short.

"Well, that's nice," you say, "How does one do this kind of programming?" I'm glad you asked! The first thing you need to do is break out your pencil and paper and start drawing pictures. Let's say that you have a mini Sumo robot that needs to do three things to stay alive on the Sumo board:

1. Look for the other guy.
2. Attack the other guy.
3. NOT fall off the edge of the board.

This seems obvious for a mini Sumo, and is a great place to start drawing pictures of your state machine. **Figure 1** shows the simplest form that this FSM can take.

One always has a first state (initialize, in this case) that gets the ball rolling for the state machine and explicitly defines where the state machine starts. This state passes off to *Search*, which then passes to *Attack*, and then to *Detect Edge* which passes back to *Search*. Each of these states represent a block of code that is in itself another state machine. Seems simple, right? Who out there sees the problem with doing this on a robot? Exactly! The motors! Who controls the motors? They can't all do it, or every state in the main loop will tell the motors to do something different, and this will happen very, very often because if you have done your job right, your code will be switching quickly to avoid *starving* any of your main states in **Figure 1**.

We now must add in the concept of priority. In the case of our mini Sumo's motors, we have a single resource that several processes want to control. Only one of them can have control at a time. The best way to handle this is to determine which of the processes is the most important, which is second, and so on. My choice is the *Detect Edge* process for most important. If we run off of the edge, the game is over and our robot looks dumb. My vote for second most important is the *Attack* process since you want it to react when the *Search* process finds something.

Now we know that we need an FSM which keeps switching between processes that are each also FSMs, and that those processes need to have a priority attached to them to gain proper access to a limited resource — the motors. How? To make everything work well, we need to break each of these processes up into bite size chunks of code that run fast, do something, and then exit after setting up their next state to process. Let's look at how we would have the Attack process do this. **Figure 2** shows a breakdown of a Search process that I have used.

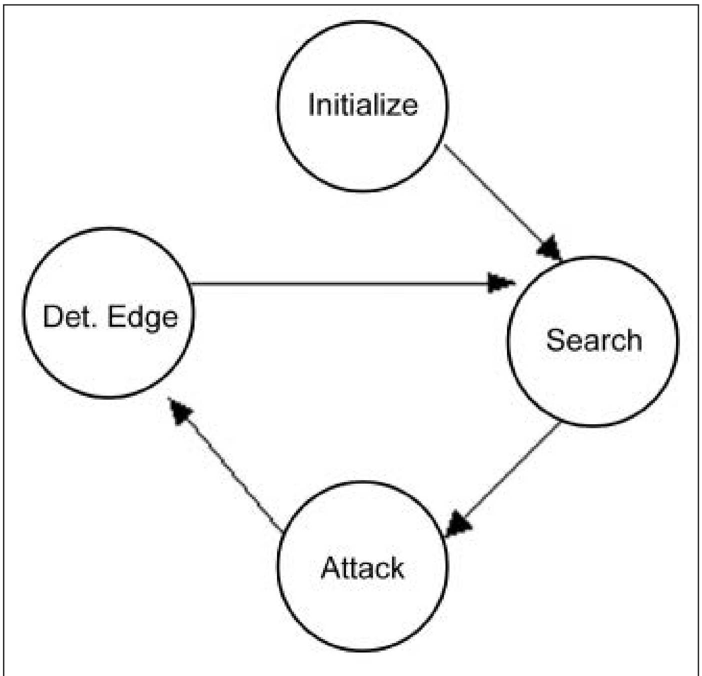
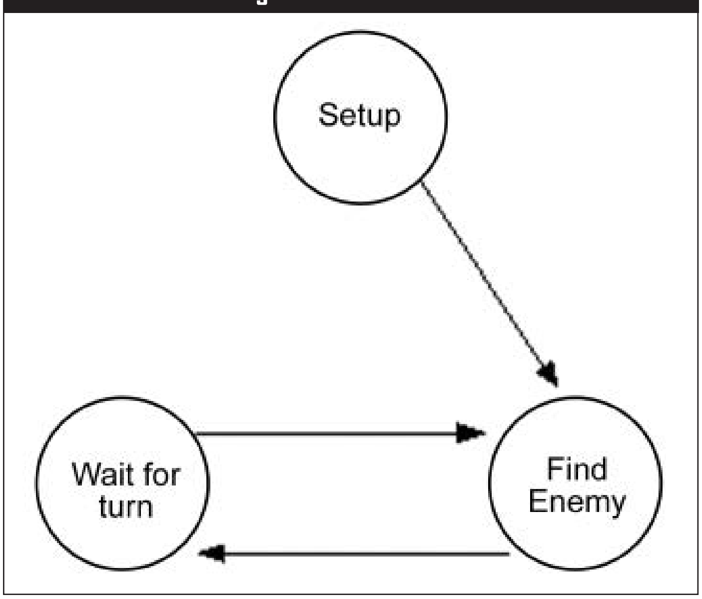


Figure 1. Simple state machine.

Seems pretty simple? It is, actually. This FSM simply looks at a sonar and determines if something is to the right or left. The *Find Enemy* state then sets up a motor direction, sets the next state, and exits. When this process is again entered, it will be in the *Wait for turn* state where it will remain until it has completed its turn. Then, it will set the state back to *Find Enemy* and start over again.

"But when do I do the sensors? When does the motor

Figure 2. Search FSM.



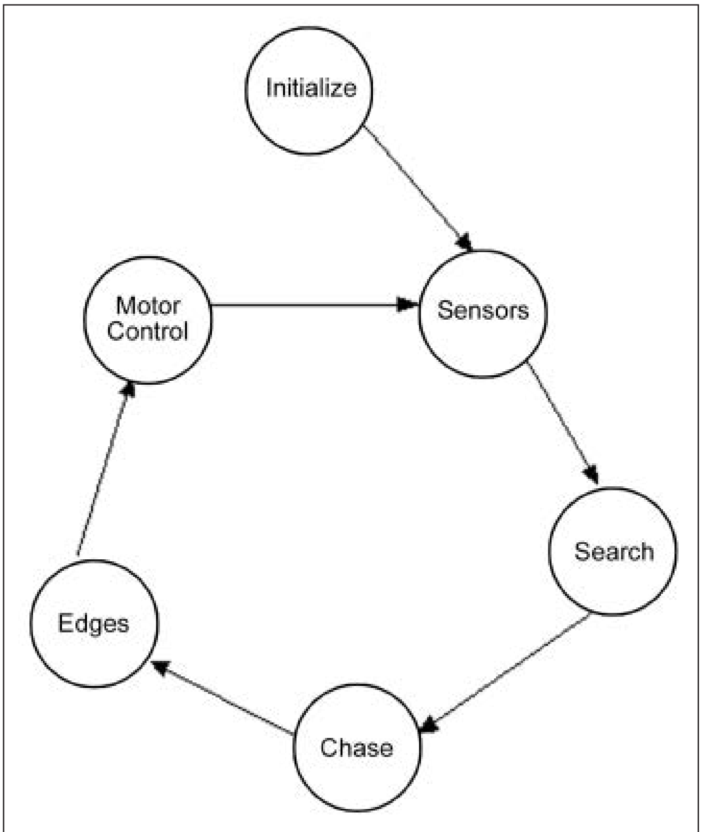


Figure 3. Mini Sumo FSM.

get turned on?" Good question! My main level FSM has states that handle that. The individual processes will handle the priority arbitration, then the motor controller state looks at the settings it has been given and acts upon them. I also have a state whose whole job is to set up and read sensors. Those values are left where the processes that need sensor input can get them and decide what to do. So, let's look at a more complete FSM that includes all of those states for a more complete picture. **Figure 3** shows a pretty good FSM for a simple mini Sumo.

Figure 4. White Stripe.

There is a wealth of information not shown here in these drawings, but this is how it's done. We draw pictures to get the main process flow down. Then, we break each of the higher level states down into their own FSMs and decide their flow. It is rare that I've gone any deeper than two levels of FSM; at that point, I'd be thinking of interrupt service routines which is how you should do timers. In many of these processes, you will be wanting to do something for a while, then stop and do something else. For instance, if your *Edges* process detects the edge of the board, it will want to back up and turn around. You do *NOT* want to sit in a for/next loop to do this! You would set the motors to reverse, set a timer, and exit. The next state would monitor the timer and switch states to the *turn* state, set the motors, set the timer, and exit, and so on.

Pictures are nice, but eventually you will need to write code. **Listing 1** shows the code that I used in a mini Sumo as a *Search* process; I called it "track()."

In this code, I used a simple counter (dur) to determine when to be done with a state. A timer or a counter are both viable ways to keep track of time. Note the beginning; if anything with a higher priority than TRACK was running, then we do nothing — we're already busy. The initial **if** block is the *initialize* phase of the state machine that gets the FSM set up and running. At that point, we use the **switch** statement to track each state of the state machine. The *leftSONAR* and *rightSONAR* variables are set in the *sensors* state of the main FSM to handle all of the various sensor inputs. Finally, the movement states set the *lS* and *rS* variables that the *Motor Control* state will use to set motor speed and directions.

This method of programming does a lot more than just allow us to have the appearance of concurrent processes using cooperative multitasking. It also allows us to break our processes down and isolate them so that we can add new ones, and change them without the risk of breaking some other part of the code. Yes, that is correct, you CS majors out there. This allows us to nicely compartmentalize our code to enhance cohesion and reduce coupling. An older book, *Mobile Robots* by Jones and Flynn, gives a great introduction into the concepts of this type of programming. These authors are alumni from the MIT robotics program run by Rodney Brooks. In this book, this type of FSM was called subsumption programming for behavioral robotics; each of the main level states would be called a behavior and have a priority attached to it. I recommend this book for some light reading.

For even more on behavioral programming I really recommend *Vehicles: Experiments in Synthetic Psychology* by Valentino Braitenberg. This simple and fast reading book will open your eyes to great ways to think of programming simple robots that make them very life-like.

No discussion about a robot would be complete

Listing 1: A Search FSM.

```
void Track(void)
/*
 * We see something, try to get closer to it.
 */
{
      static uint8_t dur=0;  //timer holder
      static uint8_t tState = 0;    //state
                                    //holder
      static uint8_t direction = 0; //which way
                                    //to go?

      if (priority > TRACK)
      {
              return;   //something higher
                        //priority is running
       }
        if ((leftSONAR < IN_SONAR) || (rightSONAR
< IN_SONAR) || (priority == TRACK))
      {
              WinkTracker();
              if (priority < TRACK)   //we trump
                                      //this
              {
                      printf("I see him\n\r");
                      priority = TRACK;
                      tState = 1;   //set up for
                                    //tracking
                                    //action
              }
              switch (tState)       //state
                                    //machine
                                    //here
              {
              case 1:               //turning
                                    //left
                      if (leftSONAR < IN_SONAR)
                      {
                              direction = LEFT;
                              lS = HALFREV;
                              rS = HALFFWD;
                              dur = 20; //is this
                                                        //long
                                                        //enough
                                                        //to turn?
                              tState = 2;
                      }
                      else if (rightSONAR <
                      IN_SONAR)
                      {
                              direction = RIGHT;
                              lS = HALFFWD;
                              rS = HALFREV;
                              dur = 20; //is
                                        //this
                                        //long
                                        //enough
                                        //to turn?
                              tState = 2;
                      }
                      else    //nothing to do, go
                              //back to search
                      {
                              tState = 0;
                              priority = 0;
                              tState = 0;
                      }
                      break;

              case 2:         //wait for turn to
                              //be over
                      if (dur == 0)
                      {
                              tState = 1;   //go
                                            //check
                                            //again
                      }
                      else
                      {
                              dur—;
                      }
                      break;
              }
      }
}
```

without a picture of the actual robot in question! **Figure 4** is my little mini Sumo named "White Stripe" after one of my current favorite Indy groups. You'll note there are no sonar units – I replaced them with IR range finders that hide inside. Another nice feature of FSM programming and breaking your code up is easy substitution of new code and hardware! Enjoy.

Here we are again at the end of another column, another month, and coming up on another year. Where did all the time go? I hope you got something from me this month, and I hope you keep sending letters with questions. I like answering them! You can reach me at roboto@servomagazine.com. I'll be happy to work on your issues! Until next time, keep on building those robots! **SV**

EVENTS Calendar

ROBOTS.NET

Send updates, new listings, corrections, complaints, and suggestions to: steve@ncc.com or FAX 972-404-0269

Know of any robot competitions I've missed? Is your local school or robot group planning a contest? Send an email to steve@ncc.com and tell me about it. Be sure to include the date and location of your contest. If you have a website with contest info, send along the URL as well, so we can tell everyone else about it.

For last-minute updates and changes, you can always find the most recent version of the Robot Competition FAQ at Robots.net: **http://robots.net/rcfaq.html**

— R. Steven Rainwater

NOVEMBER

6 **Bloomington VEX Tournament**
Ivy Tech Community College, Bloomington, IN
The Bloomington VEX Tournament includes several VEX events including Hoopla, Full Pull, and a fully autonomous event.
http://sites.google.com/site/bloomington roboticsclub

6-7 **Korea Intelligent Robot Contest**
Pohang City, Korea
The Korea Intelligent Robot Contest includes events for autonomous cleaning robots and an intelligent robot contest.
http://irc.piro.re.kr

7 **International Micro Robot Maze Contest**
Nagoya University, Japan
This Micro Robot Maze Contest will have lots of events for tiny robots including Micro robot racers, climbers, and maze solver robots; all one cubic cm or smaller. Also a two leg walker contest for two inch robots.
http://imd.eng.kagawa-u.ac.jp/maze

7 **Roaming Robots Grand Final**
Maidstone, UK
In the Roaming Robots Grand Final, remote controlled vehicles attempt to destroy each other.
www.roamingrobots.co.uk

13 **DPRG Roborama**
Dallas, TX
The DPRG Roborama will have indoor and outdoor events for autonomous robots including line following, square dance, table top, and RoboColumbus.
www.dprg.org/competitions

13 **Robotic Arena**
Wroclaw, Poland
Mini Sumo, Sumo, line following, and freestyle events.
http://lirec.ict.pwr.wroc.pl/~arena

18-19 **Real World Robot Challenge**
Tsukuba Expo Center, Tsukuba, Japan
In the Real World Robot Challenge, autonomous robots must navigate on real world streets and sidewalks.
www.ntf.or.jp/challenge

19-20 **Texas BEST Regional Championship**
UNT Coliseum, Denton, TX
This year's contest is called Total Recall.
www.eng.unt.edu/texasbest

19-21 **All Japan MicroMouse Contest**
Tsukuba, Japan
In this MicroMouse Contest, speedy autonomous robots must solve a maze in the shortest time possible.
www.ntf.or.jp/mouse

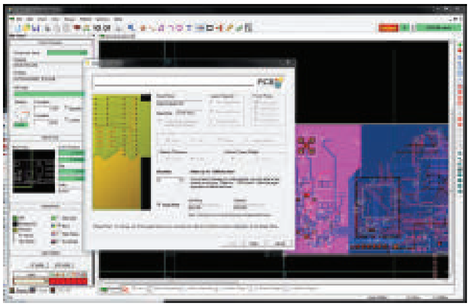
20-21 **Canadian National Robot Games**
Ontario Science Center, Toronto, Ontario, Canada
This competition has events that include line following, Sumo, search and rescue, fire-fighting, and a walker race.
www.robotgames.ca

21 **Robocon**
Tokyo, Japan
Robocon is a nationwide robotics championship event with teams from over 50 schools participating.
www.official-robocon.com

NEW PRODUCTS

DESIGN SOFTWARE

PCB123 v4 Makes Design Process Easier



Sunstone Circuits, a printed circuit board (PCB) prototype solutions provider, is launching the newest version of their design software, PCB123 v4. This advanced software is the most up-to-date, enhanced version of their free-to-use, no license required design-tool. It has an intuitive CAD interface that lets you create new PCB designs quickly, offering freedom and flexibility in your schematic and layout editing.

In their quest to remain the easiest PCB solutions provider to do business with, Sunstone Circuits has responded to feedback from the engineering community by not only improving the feature set of PCB123, but also configuring the software into a more user friendly and intuitive resource. As a result, design engineers now have access to a free industrial-grade CAD tool that is setting a new standard in the industry. Some of the key features in the release of PCB123 v4 include:

- 500,000 New Parts: Improved search functionality with complete access to parts libraries (powered by Accelerated Design).
- Datasheet Availability: Research in 'real time;' parts availability from Digi-Key.
- Automated BOM (Bill of Materials): For ease in purchasing and assembly.
- New Data Importer: Gain efficiency by importing DXF files from mechanical CAD tools, removing tedious pain-points that slow down the design process.
- Buildable Designs: Sunstone's DRC/DFM rules are integrated into the software to deliver more effective feedback and fewer design spins.

PCB CAD tools have traditionally concentrated on the hard engineering issues, like impedance calculations and trace routing tools. What has been overlooked is the ability to provide design engineers with information regarding the economic or supply-chain constraints that affect design times. With PCB123 v4, Sunstone brings the first fully-realized set of tools that allows its users to design with full knowledge of the budgetary and scheduling impacts of their design decisions, as well as the hard engineering information. Sunstone realizes that their users are held accountable for financial and budgetary constraints and that the tool they use should provide them that information.

"With PCB123 v4 and our new approach to making parts libraries available, we're tackling one of the most overlooked aspects of PCB design: parts management. It's estimated that PCB design teams spend at least two million resource hours annually just maintaining parts libraries in existing tool sets. Most of that work is duplicated in numerous other design teams. Our work with Digi-Key helps buy all that time back for PCB designers to do what they do best — design PCBs," — explains Nolan Johnson, CAD/EDA Manager, Sunstone Circuits.

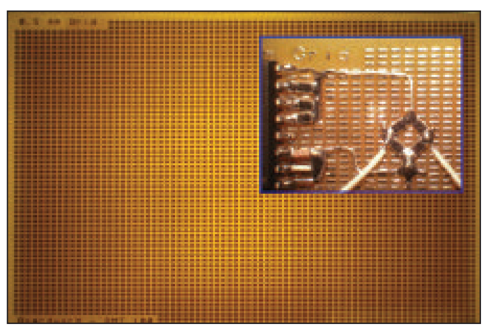
In addition to the enhanced relationship with Digi-Key, Sunstone Circuits has developed a new partnership with parts-libraries supplier, Accelerated Designs. With their Ultralib product, their parts libraries are available to users at no cost when inside this design tool environment.

For further information, please contact:

Sunstone Circuits — Website: **www.Sunstone.com**

PROTOTYPING

Unique System for Surface Mount Prototyping With SMT Packages



BoardworX — from MX2 Systems Corp. — is a new SMT prototyping system that allows circuit builders to use the latest SMT parts when creating new designs in a convenient and easy to use form. BoardworX utilizes a "universal grid" concept which accommodates mounting of all standard packages

available for SMT parts currently available (except BGA) and applies simple point-to-point wiring to achieve a quick prototype circuit. The system removes the delay in layout, fabrication, and delivery of a custom PCB and provides a general platform on which almost any component can be mounted in many orientations. It can be considered "perf board for surface-mount."

The system consists of a simple grid of pads that is designed to accommodate all pin pitches available on various parts by having the user shift the device around at mounting time to find the best fit, then tacking the part down. This is followed by hand-wiring the interconnects between parts with bare fine wire. The result is a small, tight neat package, utilizing SMT and through-hole parts if desired.

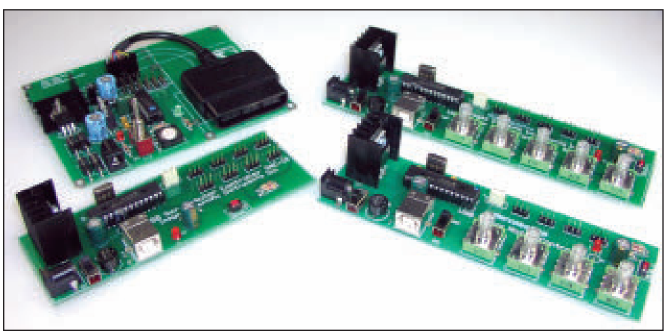
The series currently has four board choices which allow for different construction capabilities. All boards except the SMT-300 have different grid spacing on either side of the board for the best fit of components. The SMT-100 is a basic board, 2 x 3 inches, featuring a universal grid on which to mount parts; it's intended for general prototyping of small to medium sized circuits. The SMT-200 features a set of large pads on the back of the board that can be used as a plane for convenient electrical connection (via holes through the board) or for heatsinking and mounting larger power components. The SMT-300 addresses the SMT and through-hole hybrid designs, combining 100 mil spaced through-holes which border two SMT universal grid areas. Additionally, this board can be used for building custom adapter boards from surface-mount parts for mating to solderless breadboard applications. The SMT-400 is a small mini board (just under one inch square) allowing for small "fix" or daughterboards to be constructed.

For further information, please contact:

MX2 Systems Corp.

Suite 20053, Town Centre
Kelowna B.C. Canada
V1Y 9H2
Tel: **877 • 471 • 9886**
Email: **info@BoardworXsystem.com**
Website: **www.BoardworXsystem.com**

MOTOR CONTROLLERS



Integrated Power Supply

Images Scientific Instruments, Inc., new IPS (Integrated Power Supply) is setting a new standard of excellence in servomotor controllers. This line of servomotor controllers allows you to use a variety of inexpensive power supplies to run both your controller board and servomotors. Use available power supplies, transformers, batteries, wall transformers, etc. — anything from 6V to 30V, either AC or DC.

The IPS system will regulate the power to run your servomotors efficiently. Power is supplied through a 2.5 mm power socket on board. A discrete 2.5 mm power jack is provided with the controllers for connecting your own power supply.

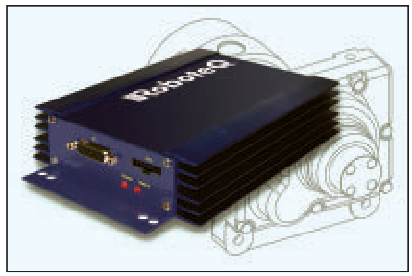
Pictured here are the newest products in their line of servomotor controllers, starting at the upper right and moving clockwise: the PS2-SMC-06, USB-SMC-05, USB-SMC-4, and USB-SMC-08.

For further information, please contact:

Images Scientific Instruments, Inc.

Website: **www.imagesco.com/servo/motion-control-index.com**

Smart 7.5 KW Brushless DC Motor Controller



Roboteq, Inc., introduces an intelligent controller capable of directly driving a brushless DC up to 150 amps at up to 50V. The BL1650 is targeted at designers of electric bikes or karts, mobile robots, or any other high power brushless motor control application.

The controller accepts commands from either analog pedal/joystick, standard R/C radio for simple remote controlled robot applications, or RS-232 interface. Using the serial port, the BL1650 can be used to design fully or semi-autonomous robots by connecting it to single board computers, wireless modems, or WiFi adapters.

The BL1650 incorporates a Basic language interpreter capable of executing over 50,000 Basic instructions per second. This feature can be used to write powerful scripts for adding custom functions, or for developing automated systems without the need for an external PLC or microcomputer.

The BL1650 uses the motor's Hall sensors to measure speed and distance travelled with high accuracy. The controller can operate the motors in open loop or in closed loop speed, or position mode with a 1 kHz update rate.

The BL1650 features intelligent current sensing that will automatically limit the power output to 150A in all load conditions. The controller also includes protection against overheat, stall, and short circuits.

For further information, please contact:

RoboteQ

Website: **www.roboteq.com**

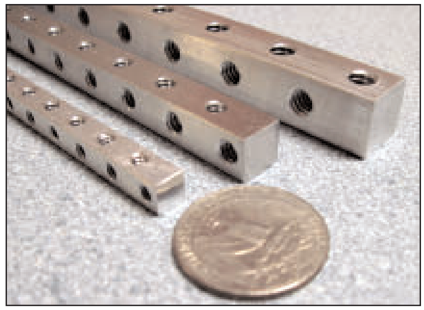
TOOLS & ACCESSORIES

Mount for 28 mm Brushless Motor

This mount from KITBOTS allows the easy mounting of a 28 mm brushless inrunner motor to a flat panel. Machined from tough 6061 aluminum, it clamps the motor firmly but gently. All required screws are provided. This part provides a simple way to add a powerful weapon drive motor to your Beetleweight (3 lb) combat robot.

Nutstrip Makes Joining Panels Easy

This Nutstrip also from KITBOTS allows the quick and easy joining of two panels at right angles to each other.



It is cross-drilled and tapped so it can replace the use of brackets and nuts. Machined from tough 6061 aluminum, it is available in three sizes from 1/4" to 1/2" square. It's ideal for building a chassis or attaching armor to your robot.

For further information on these two items, please contact:

KITBOTS	Website: **www.kitbots.com**

Is your product innovative, less expensive, more functional, or just plain cool? If you have a new product that you would like us to run in our *New Products* section, please email a short description (300-500 words) and a photo of your product to:

newproducts@servomagazine.com

bots IN BRIEF

STAYING IN (iPod) TOUCH

Robovie mR2 is a 30 cm tall humanoid robot developed at Japan's ATR Robotics and Communication Laboratories. It sports 18 joints (three in each eye, three in the head, four in each arm, one in the back), 18 servo motors, CCD sensor 3.4 MP camera, two microphones, and a mono speaker (2W). The real trick is the communication channel is established through an iPod Touch that you need to place into its chest. It is not only possible to control this little companion by touching the screen of the iPod Touch but also via Wi-Fi (802.11b/g) or Bluetooth. According to ATR, it can communicate the info from your iPod Touch to you by his gestures. Unfortunately, it is not commercialized yet. However, ATR states they are considering that.

GIVE THEM A HAND

Launched at the International Society for Prosthetics and Orthotics (ISPO) in Leibzig, Germany by the UK-based company Bebionic, this myo-electric hand fully integrates a wireless chip, 19 different silicone skin tones (so the skin tone of the hand can easily blend in with the user), as well as fully customizable functions to control speed, grip force, and grip pattern. It supersedes past mechanical hands which were capable of flexion/extension whereas this is reportedly capable of rotational articulation.

This myo-electric hand is so sophisticated that you can play piano with it, play Jenga, and even do sign language. You can also grasp various objects such as a can of soda, a bouncy ball, and wooden blocks.

MAKE MINE A DOUBLE

Joining Tosy on the Vietnam bandwagon is their first humanoid waiter robot " Topio Dio" who was unveiled at Automatica 2010 (International Trade Fair for AUtomation and Mechatronics) in Germany. Developed as a service robot — or as so-called "Skillful Bartender and Waiter" — Topio Dio is 125 cm tall and weighs 45 kg. It is equipped with 28 joints, three wheels, a built-in camera and sensor, and is controlled via Wi-Fi. Although the price has not been disclosed, apparently it is only about 25% of the cost for similar products on the market.

bots IN BRIEF

EMIEW2 IS BETTER THAN (FIRST) ONE

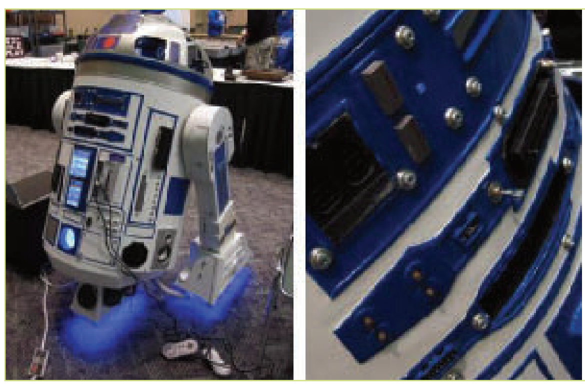
EMIEW2 is actually Hitachi's new and improved second-generation EMIEW humanoid robot. They developed EMIEW in 2005 followed by the EMIEW2 in 2007 with the objective of creating a robot that can coexist with humans and serve in such environments as offices or hospitals as part of the Project for the Practical Application of Next-Generation Robots organized by NEDO (New Energy and Industrial Development Organization).

The first one was 130 cm tall and weighed approx 70 kg while EMIEW2 stands 80 cm high and weighs just 13 kg. The new version still moves at the average human walking speed of 6 km/h. The improvements include better recognition of voice commands (thanks to a 14-channel microphone system positioned within its head), smoother movement on uneven surfaces, and better handling of bumps up to 1.5 cm high thanks to the new "active suspension system."

SHALLOW PAL

Apparently, South Korea's Ministry of Land, Transport, and Maritime Affairs is planning to invest 20 billion won in an 'underwater robot' project until 2015. The project is about the construction of an "articulated, multiple-mobile" robot which will be capable of handling various tasks — at a maximum of 6,000 meters underwater — ranging from exploration and rescue operations to such environmental missions as maintenance of underwater flora. The development of the robot with capabilities to operate in shallow water is the first phase of the project and is scheduled to be completed by 2012. During the second phase from 2013 to 2015, the robot will further be improved to perform its planned deep water tasks.

I'M STUFFED!



In this R2D2, you'll find a Nintendo Entertainment System, a Super NES, a Nintendo 64, a Nintendo GameCube, a Playstation 2, a Sega Genesis, a Sega Dreamcast, an Atari 1800, an XBox 360, and a computer, plus parts of a PS1 and an XBox 1, AND a PSP hidden in one of the legs. What's really cool is how all of the switches, buttons, and ports for controllers and stuff have been distributed all over the case and integrated into the design.

Lots of people build R2s based on the original ILM specifications, but there's usually a compromise between various degrees of functionality because it seems to be more or less physically impossible to stuff a lot inside the case of an R2. That doesn't seem to be the case here. Apparently, the hardest thing to do is a 3-2-3 conversion which is getting R2 to extend and retract a functional center wheel. If you're interesting in building your own R2, there's tons of information at **www.astromech.net** which is the official site of the R2D2 Builder's Club.

Cool tidbits herein provided by Evan Ackerman at www.botjunkie.com, www.robotsnob.com, www.plasticpals.com, and other places.

ARCHER RIGHT ON TARGET

This robot only took eight trials to figure out how to hit the center of a bullseye. iCub is using a learning algorithm called ARCHER (Augmented Reward Chained Regression) which is optimized for tasks that have an easily definable goal and measurable progression towards that goal. Basically, hitting the center of the target equates to a maximum reward, and the algorithm builds off of past experience to estimate how to alter iCub's hand positions to improve the aim of the arrow. In this case, the distance between iCub and the target is only 3.5 meters, but there's no reason it couldn't be scaled up to larger distances.

This robot experiment was conducted by Dr. Petar Kormushev, Dr. Sylvain Calinon, and Dr. Ryo Saegusa at the Italian Institute of Technology (the same guys who brought us robot pancake flipping).

ARCHER uses a chained local regression process that iteratively estimates new policy parameters which have a greater probability of leading to the achievement of the goal of the task, based on the experience so far. (Huh?) An advantage of ARCHER over other learning algorithms is that it makes use of richer feedback information about the result of a rollout.

For the archery training, the ARCHER algorithm is used to modulate and coordinate the motion of the two hands while an inverse kinematics controller is used for the motion of the arms. After every rollout, the image processing part recognizes automatically where the arrow hits the target which is then sent as feedback to the ARCHER algorithm. The image recognition is based on Gaussian Mixture Models for color-based detection of the target and the arrow's tip. iCub has 53-DOF and is 104 cm tall.

This research will be presented at the Humanoids 2010 conference December 6-8, 2010, in Nashville, TN.

Ronald Arkin and Alan Wagner

TELL ME SWEET LITTLE LIES

A robot deceives an enemy soldier by creating a false trail and then hiding so that it will not be caught. While this sounds like a scene from one of the Terminator movies, it's actually the scenario of an experiment conducted by researchers at the Georgia Institute of Technology as part of what is believed to be the first detailed examination of robot deception.

"We have developed algorithms that allow a robot to determine whether it should deceive a human or other intelligent machine, and we have designed techniques that help the robot select the best deceptive strategy to reduce its chance of being discovered," explained Ronald Arkin, a Regents professor in the Georgia Tech School of Interactive Computing.

Can robots lie? This is the subject of the research that enables them to detect whether one is susceptible and use that gullibility against another based on algorithms. They feel that the deception could help it avoid being captured by evil military sources and calm those they rescue, but robotic experts claim that this ability would not only damage robot's current positive image, but may lead to teaching them to be able to gamble and hunt.

SURVIVAL GAMES PART DEUX

The first ever Robot Survival Game was just barely a couple months ago, but everybody managed to patch up their robot's tinfoil hats for round two.

In the second Robot Survival Game, eight games took place such as:

Extermination Fight (destroy the enemy completely): Each team fights to destroy the enemy completely within five minutes. The winning team has the larger number of surviving robots after five minutes.

Flag Fight: Each team fights to shoot down the flag in enemy territory. The winner is the team that shoots down the enemy flag within five minutes, or the team which has the larger number of surviving robots after five minutes if a flag is not shot down.

Duel Fight: Two robots battle it out in the field.

TAKING CHARGE

Panasonic's Evolta batteries set a new world record last year for travelling 3.726 km at LeMans, France.

This year, Panasonic will try to set another world record by travelling from Tokyo to Kyoto (that's 500 km/310 miles) with its new robot ROBO-GARAGE. The 53 stations of Tokaido on the road will serve to recharge the batteries. The "race" started September 23rd and will continue through December 10th.

You can follow this adventure on the official website at **http://panasonic.jp/drycell/evolta/challenge/53**/.

The new Evolta batteries became available October 1st.

WHAT THE HOALOHA?

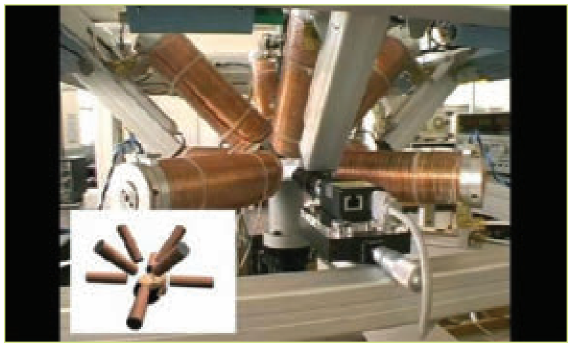
Tandy Trower, who helped launch Microsoft Robotics Studio back in 2006, has started a new robotics company called Hoaloha Robotics. The goal? Affordable ($5,000–$10,000), socially assistive (i.e., elder care) robots in the next three to five years. Trower envisions a robot able to do all of the conventional remote monitoring and pill reminder stuff, but also be able to assist with movement, object retrieval, and potentially provide some degree of intelligent social interaction.

Trower believes he can make an important contribution by developing a common interface and software that will make assistive robots easy to use and customize with applications — similar to the way Apple standardized the interface and application model for smartphones. "This is what primarily I believe is holding back most of the industry right now. It's not that robots can't be built, it's that nobody has defined the software that's going to turn robots into useful appliances," Trower commented.

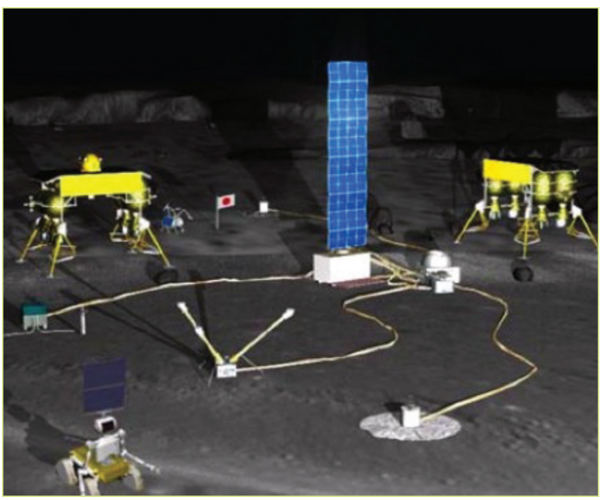
"The components exist; it's not difficult to build such a platform," he continued. "What people have lacked is the ability to envision what the right package should contain and, most important, what the applications and user interface should be."

In order to be an effective assistive robot, the Hoaloha platform is going to have to be very independent. This specifically is what Hoaloha is going to be focusing on, partnering with other companies for hardware development.

I SEE



See this "friendly" looking piece of equipment? Just stick your head in the middle and a tiny robot will fix your eye problems. Those giant coils are very precise magnetic field generators, and they're capable of manipulating a half-millimeter long microbot with a fine enough touch to get it to fix clots in blood vessels in your eyes. The whole system is called OctoMag, and the primary advantage that it offers (besides the robotic fine manipulation that makes other assistive surgical systems so promising) is that the microbot is completely untethered. So, instead of having to shove a bunch of needles into your eyeballs and dig around, one single needle can deposit the robot which does its thing with minimal invasiveness and then comes back out via the same needle.



TO THE MOON

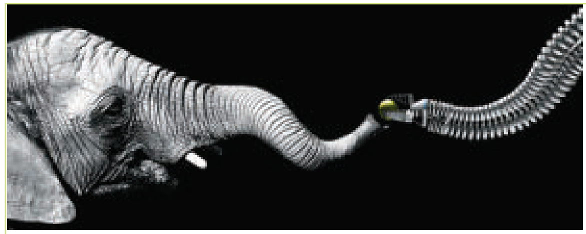
Japan is planning to invest 200 billion yen for a robotic base near the south pole of the Moon by 2020. For the project, rover robots (some of which weigh about 300 kg) will be exploring the Moon's inner structure and gather rock samples. In order to accomplish these tasks, they will be equipped with solar panels and seismographs, as well as HD cameras and arms. The government is also planning to establish a power system based on solar power and lithium-ion batteries. Another interesting detail is their plan for an HDTV broadcast from the Moon.

This unmanned lunar base is basically a precursor for Japan's manned lunar base project scheduled to start in 2030.

Japan also plans to send a humanoid to the moon by 2015.

According to the Osaka edition of the YomiuriOnline newspaper the "Lacuna Oriented Higashiosaka Important Conjunction," there's a plan to build a two legged robot, Madio-Kun, or humanoid to be fully operational for space travel and specifically to be operated on the surface of the Moon by 2015.

WORKING FOR PEANUTS



This is that elephant trunk robot arm thing from Festo.

Festo's design is actually a clever halfway house between a fully-articulated robot arm that mimics the musculo-skeletal structure of a human arm and a basic rotating-joint robot arm like the kind you might find currently in action welding car parts together on a production line. It's intended to perform manipulator tasks like gripping objects off a production line and popping them into boxes, and it looks to be more dextrous than a simple jointed arm, while also being more delicate and clever about gripping the objects it finds. That would seem to make it ideal for manufacturing tasks that current robots might not be so suitable for, such as in the food industry.

The Festo arm is based on an elephant's trunk anatomy with segmented parts that are made of lightweight rapid-prototyped plastic material and air pressure-driven "muscle" chambers.

LEAN, MEAN, HRP-4 MACHINE

Kawada Industries and the National Institute of Advanced Industrial Science and Technology (ASIT) have unveiled the latest edition to their family of androids: the HRP-4. HRP-4 is designed "in the image of a lean but well-muscled track-and-field athlete." It is pretty damn lean — at five feet tall, it only weighs 86 pounds and it boasts increased flexibility of its 34 joints over its predecessors. Despite its apparent lack of big fat heavy stuff like powerful motors, computers, and batteries, it has no trouble doing all of the important android basics.

HRP-4 is designed to aid in the development of robots that could replace humans in simple manual labor, specifically to address Japan's impending labor shortage (due to an aging population and low birthrate).

HRP-4 will be available in 2011 for about $305,000.

SWAN SONG

This is a robot swan. A Swedish robot swan, in fact. This particular Swedish robot swan has been taught to dance as part of a collaborative project between the computer science and theater departments at Mälardalen University. The actual choreography was done by professional dancer Asa Unander-Scharin, who programmed movements into the swan's wings, neck, beak, and feet. Apparently, the performances move people to tears, and spectators have described the experience as "touching," "fascinating," and "beautiful."

COMBAT ZONE

Featured This Month:

PARTS IS PARTS

TAIG Tools Desktop CNC Mills

● by Kevin M. Berry

On the ever interesting Robot Fighting League forum, I ran across this spontaneous product review. It meets my criteria for this column, which is "builder tested." I am a firm believer that all product reviews should be after they've been "in the box." For tools, the combat arena is a builder's shop, of course! To set the stage, here's my favorite builder/disaster/shop quote of the month: "I tried TIG welding Ti in a box filled with Argon. The box caught on fire."

Now that we've baselined the "stress tests" that a builder can inflict on shop equipment, here's the request from a builder:

"I just checked out the Taig Tools website and fell in love with the CNC mill (**www.taigtools.com/cmill.html**). Anyone have any experience with this mill?"

Two experienced builders weighed in. One commented, "I have one. It's a few years old, so it is an open loop stepper made by Micro Proto. It's the four axis machine running Mach 3 software. Mods I've made include a custom fixture plate, 18 inch table, and 1" Y extension. It is fairly rugged and rigid, not to mention pretty accurate. I've been able to mill 0.1 depth steel with a 1/4" end mill. It's light enough to lift and place anywhere." (Editor's note: The website shows the CNC version


CNC mill.

now uses closed loop servo digital sync lock servo technology. There is a lower grade "CNC converted" version at about half the price of the advanced CNC version.)

Chris Baron of Robot Power went into more detail:

I've had one of these for years. Got mine from Super Tech using their conversion and controller box. I cut mostly Al but I cut a saw blade to make a replacement firing pin for an old .22 once. Tried to cut Ti once but it was a disaster. Couldn't get the heavy feed needed to cut. I was probably doing it wrong but it is much more difficult to cut than Al, plastic, or steel where you just set the feed speed.

Works well overall as long as you remember it is a light duty machine for small parts. I find the Y axis travel to be frustratingly small. Z and X are fine (I have the 18" table).

They are a pain to keep lubed. I use an old-time oil can with a squeeze handle and a flex tube to reach the ways and screws. I use ATF fluid for the lube. Way oil is way hard to find in small bottles and/or locally.

I find them to be adequate in terms of speed and accuracy. The spindle motor on mine is weak for heavy feeds. The newer ones have larger spindle motors but I haven't upgraded. (Editor's note: According to their website, they now offer a 1/4 hp, continuous duty, 3,400 rpm motor on all CNC mills).


Servo mill.

The optimum cutter for my machine is 1/8". Smaller and they break easily. Larger and the spindle lugs down unless the speed is turned way down. Be sure to pay the extra for the TiAlN coated cutters. They are really worth it.

If I were buying a new machine, I would probably pay the extra and get a low-end Tormac. But for small parts, this will work fine. You will want to get a set of clamps for sure. There are a couple of places that make them sized for this machine. The Taig vise is only okay. I got a nice toolmaker's 3" vise from Enco that is more solid and easier to clamp stuff with.

Like all product reviews, your mileage may vary. I look forward to further input from users of this product. SV

MANUFACTURING: RioBotz Combot Tutorial Summarized — Tooth Design

● Original Text by Professor Marco Antonio Meggiolaro; Summarized by Kevin M. Berry

*Editors Note: Professor Marco Antonio Meggiolaro, of the Pontifical Catholic University of Rio de Janeiro, Brazil, has translated his popular book, the RioBotz Combot Tutorial, into English. As in previous editions of the Combat Zone, portions of the tutorial are summarized. In this article, we present a much simplified version of the "Tooth Design" section of Chapter 6 — a major treatise on combot weapons. Marco's book is available free for download at **www.riobotz.com.br/en/tutorial.html**. For a hard copy purchase (at no profit to Marco) go to Amazon. All information here is provided courtesy of Professor Meggiolaro and RioBotz.*

Tooth Design

One important issue when designing spinning weapons such as disks, bars, drums, and shells is regarding the number of teeth and their height. Too many teeth on a spinning disk, for instance, will make the spinner chew out the opponent instead of grabbing it to deliver a full blow. Everyone who's used a circular saw knows that fewer teeth means a higher chance of the saw binding to the piece being cut — which is exactly what we want in combat.

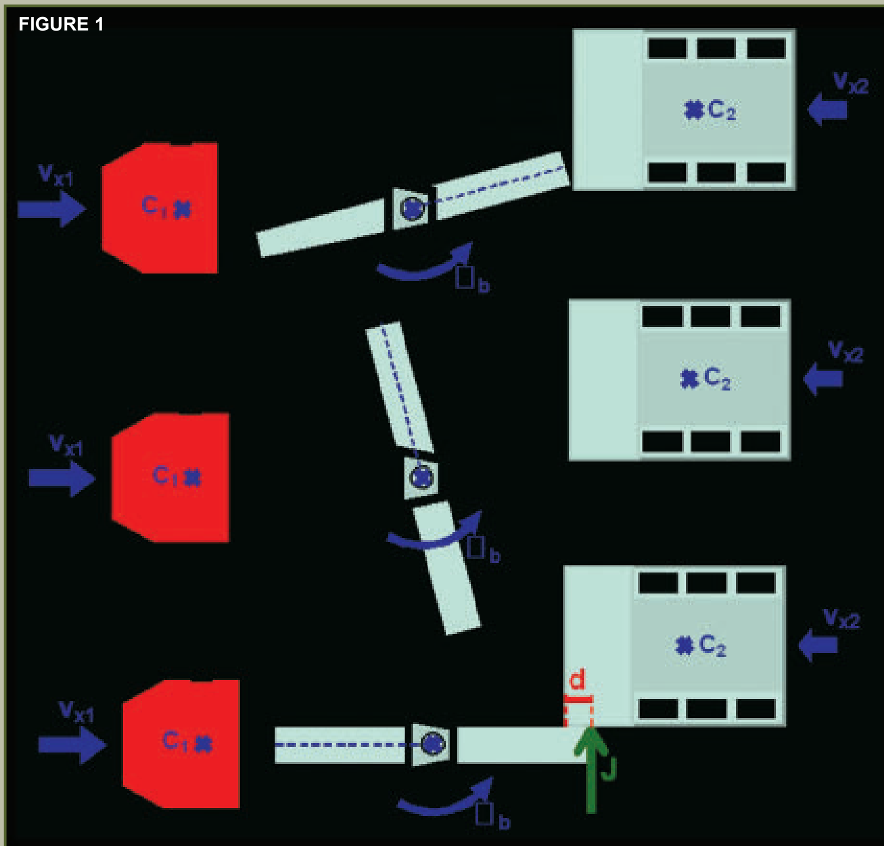


FIGURE 1

Tooth Height and Bite

Before we continue this analysis, we need to define the term "tooth bite," shown as "d" in **Figure 1**. The tooth bite is a distance that measures how much the tips/teeth of the spinner weapon will get into the opponent before hitting it. For instance, if two robots are moving towards each other with one of them having a spinning bar (as pictured in **Figure 1**), then the highest bite would happen if the bar barely missed the opponent before turning 180 degrees to finally hit it. So, the tooth bite is the "overlap" caused by the bots moving towards each other.

Small values for "d" mean that the spinner will have a very small contact area with the opponent, most likely chewing its armor instead of binding and grabbing it. So, a spinner needs to maximize d to deliver a more effective blow. This is why an attack with the drive system at full speed is more effective, since a higher closing speed will result in a higher d. This is also why very fast spinning weapons have a tough time grabbing an opponent, since their very high rotational speed ends up decreasing the tooth bite.

The maximum obtainable tooth bite can be generalized for any toothed weapon (**Figure 2**). A detailed mathematical proof is in the tutorial, but it sums up that there is no point in making the tooth height "y," any longer than the bite distance d. Intuitively, this makes sense, because there is no point in having the teeth any longer than necessary. The tooth bite should not be higher than the maximum value of d since that would decrease its strength due to higher bending moments.

The tooth height can still be reduced if necessary without compromising much of the tooth bite. This is because the equations for the tooth bite in the tutorial assumed that one tooth barely misses the opponent, until the next tooth is able to grab it. But, if instead of barely missing the opponent the previous tooth had barely hit it, it would have hit it with a distance much smaller than d. It is a matter of probability; the tooth bite can be any value between 0 and d, with equal chance (constant probability density). So, in 50% of the attacks at full speed the travel distance will unluckily be between 0 and half of d. In the other 50%, it will luckily be between half and the maximum bite distance. An (unlucky) hit with tooth bite very close to zero probably won't grab the opponent, and it will significantly reduce the attacker's speed until the next tooth arrives, decreasing the bite distance of subsequent hits. If the opponent's speed gets down to zero without the weapon grabbing the opponent, you'll probably end up grinding it. If this happens, the best option is to back up, and then charge again trying to reach maximum velocity and hoping for better luck with a high bite distance.

FIGURE 2

The chance of d being exactly at the

perfect, maximum distance is virtually zero. So if you want, you can make the tooth height shorter than the optimum dimension shown in the tutorial. If you choose, for instance, y = half of optimum, your robot won't notice any difference with this lower height in 50% of the hits since you came in on the short side of the 50/50 odds. On the other 50% (where tooth bite would be higher than half of optimum), the opponent will touch the body of the drum/disk before being hit by a tooth, resulting in a tooth bite equal to the tooth height. Designing y less than half of optimum is not a good idea, as most attacks will end up touching the drum/disk before the tooth.

For instance, the 2008 version of our featherweight Touro Feather had a drum with n = 2 teeth (**Figure 3**) spinning up to 13,500 rpm. Since the robot's top speed is 14.5 mph, then maximum bite distance = 14 mm (calculations shown in the tutorial). Since the overall height of the drum needed to be smaller than 100 mm (4") by design, a tooth height of 14 mm would result in a drum body with low diameter (72 mm or 3"). We then chose y = 10 mm for the tooth to stick out of the drum body. This 10 mm height is usually enough to grab an opponent. Also, in 10 mm/14 mm = 71% of the hits at full speed, the tooth height y will be higher than the tooth bite d. The opponent will only touch the drum body in the remaining 29% of the hits, when the next tooth will be able to hit the opponent with its full 10 mm height (unless the opponent had bounced off immediately after hitting the drum body).

Beware of a frontal collision between two vertical spinning weapons because the opponent may be able to grab your drum or disk body with its teeth before you can grab it. In this case, it is a game of chance. The robot with higher teeth will have a better chance of grabbing the opponent, as long as it spins fast enough. Since a vertical spinning bar does not have a round inner body, it basically behaves as if its tooth height y was equal to the bar radius. So, usually a powerful vertical bar will have an edge in weapon to weapon hits against drums or vertical disks.

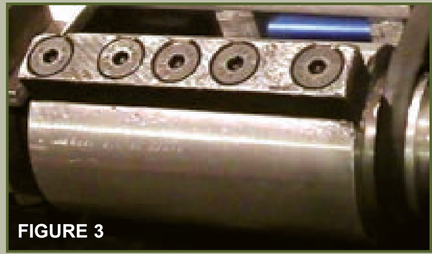


FIGURE 3

Number of Teeth

An important conclusion from the tooth bite equations is that you must aim for a minimum number of teeth. The lower the number of teeth, the higher the value of the bite distance. Disks with three or more teeth are not a good option. The best choice is to go for two teeth, as with bars or two-toothed disks. Even better is to try to develop a one-toothed spinning weapon, such as the disk of the vertical spinner Professor Chaos. However, but this requires a careful calculation to avoid unbalancing by using (for instance) a counterweight diametrically opposite to that tooth.

Note that a one-toothed weapon does not have to be too much asymmetric, nor will it need heavy counterweights, if you do your math right. For instance, the one-toothed bar pictured in **Figure 4** can be made out of a symmetrical bar, as long as the short end is chamfered properly. (Editor's note: Another long, interesting, but hard to type series of mathematical equations was deleted here and substituted with the word "properly." See the source document if you want to do the math yourself.) In this way, with the bar at full speed, even if the long end barely misses the opponent, the short end won't touch it because during a half turn, it would approach — at most — half of the optimum bite. After the full turn, it would have approached up to the best bite distance, hitting for sure with the long end. With just one tooth, it is possible to move twice as much into the opponent before hitting it, transferring more impact energy.

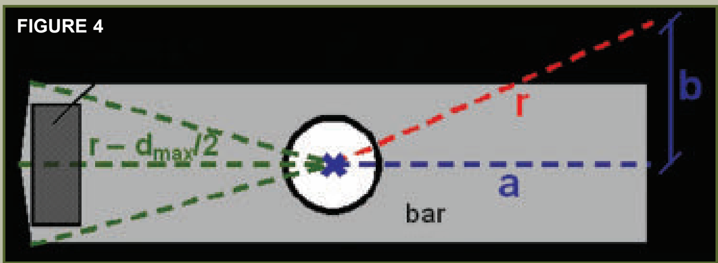


FIGURE 4

With this proposed one-toothed bar geometry, the counterweight wouldn't have to be very heavy because its mass would only have to account for the mass of the tip insert plus the removed mass from the chamfers. This bar is also relatively easy to fabricate, with very little material loss. In fact, for wide bars with large inserts — which increase the value of b — it is even possible to design the bar in a way it's almost symmetrical even after chamfering. In addition, if you perform some shape optimization removing some material from the long end, it is even possible to remove the counterweight, being careful not to compromise the bar strength at its most stressed region.

In our experience, to bind well to the opponent, the tooth bite should not be below 1/4", no matter if the robot is a hobbyweight or a super heavyweight. We've tested different tooth heights with our drumbot hobbyweight Touro Jr and featherweight Touro Feather, and values below 1/4" made the

Number of Teeth n	Maximun Drivetrain Speed	Maximun Rotational Velocity to Avoid Grinding
1	5 mph (8 km/h)	3,520 rpm
	10 mph (16 kn/h)	7,040 rpm
	15 mph (24 km/h)	10,560 rpm
2	5 mph (8 km/h)	5,280 rpm
	10 mph (16 kn/h)	10,560 rpm
	15 mph (24 km/h)	15,840 rpm
3	5 mph (8 km/h)	10,560 rpm
	10 mph (16 kn/h)	21,120 rpm
	15 mph (24 km/h)	31,680 rpm

TABLE 1

robot grind instead of grab the used dead weights. With this in mind, it is possible to generate a small table with estimated maximum weapon speeds to avoid the grinding problem. Not going over the maximum rotational velocities shown in **Table 1** guarantees that, in at least 50% of the hits at full speed, the tooth will be able to travel at least 1/4". Of course, these are just rough estimates because tooth sharpness and armor hardness also play a role helping or avoiding dents that bind with the opponent.

Summary

In this mere 1,600 words, I've managed to grossly simplify the treatment of this subject in the Tutorial. The detailed calculations and a more thorough explanation of the concepts are available for those wanting the gory details. SV

COMBAT ZONE'S GREATEST HITS

● by Kevin M. Berry

Thomas Kenney of MH Robotics (**http://mhrobotics.com/**) sent in his story.

"This is the only real 'hardcore' damage I've received to my bots as of yet. It was a beetle fight between my bot 'Cloud of Suspicion' and 'One Fierce Lawn Boy.'

Cloud of Suspicion's bottom armor is meant to be taken off between matches to charge batteries and turn the bot on and off. As a result, it's held on with a measly four 6-32 x .625" screws. This design flaw allowed Lawn Boy's drum to catch onto the inside of the carbon fiber plate after cutting through the UHMW that it's recessed into. Although the spray of robot guts was spectacular, there was no real structural damage, and the only internal components that were lost was one LiPo cell that was sliced through (and tossed into a LiPo sack right afterwards!) and the receiver. The motors that were disconnected from their gearboxes still run fine; the grazing impact of Lawnboy's weapon teeth on the motor cans had just been enough to break the grip of the blue loctite and loosen the screws." SV

Before and after photos, brief description of the fight, and builder's name can be submitted to me at LegendaryRobotics@gmail.com. Or, if you have an action shot that clearly shows what's going on, those are welcome too! These don't have to be current, anything you can (legally) submit, clear back to the good old days of wooden bots and iron builders is fair game.

EVENTS

Completed Events August 9th – September 14th

Gulf Coast Robot Sports 6 was presented by Gulf Coast Robot Sports in Bradenton, FL, on August 8th.

Robot Battles 39 was presented at DragonCon 2010 in Atlanta, GA on September 5th and 6th. SV

EVENT REPORT: Clash of the Bots – Schiele Museum 2010

● by Pete Smith

Walk in the front door of the Schiele Museum (**www.schielemuseum.org**) in Gastonia, NC and the first thing you see is a full size replica of the skeleton of a T-Rex! Fearsome though it is, there was a much more modern form of "survival of the fittest" at their annual Clash of the Bots this summer.

Carolina Combat (**www.carolinacombat.com**) worked with the Museum to put on a combined insect weight robot combat and Bot Hockey event.

This event reflects a growing trend of combining a combat robot competition with a general robot themed day at a local museum. Carolina Combat provided and set up both the Insect combat and the Bot Hockey arenas (**Figure 1**) while the Museum provided the space, nice big trophies, catering for the competitors, and crowd control.

Two teams entered for the Bot Hockey (**Figure 2**) and they fought a "best of seven" games series throughout the day with additional games and demonstrations which included members of the audience driving the bots.

Team Pneusance defeated Team Scotch Pies in four closely fought games to take the trophy. The games and the demos were very well received by the audience throughout the day.

The combat

FIGURE 1

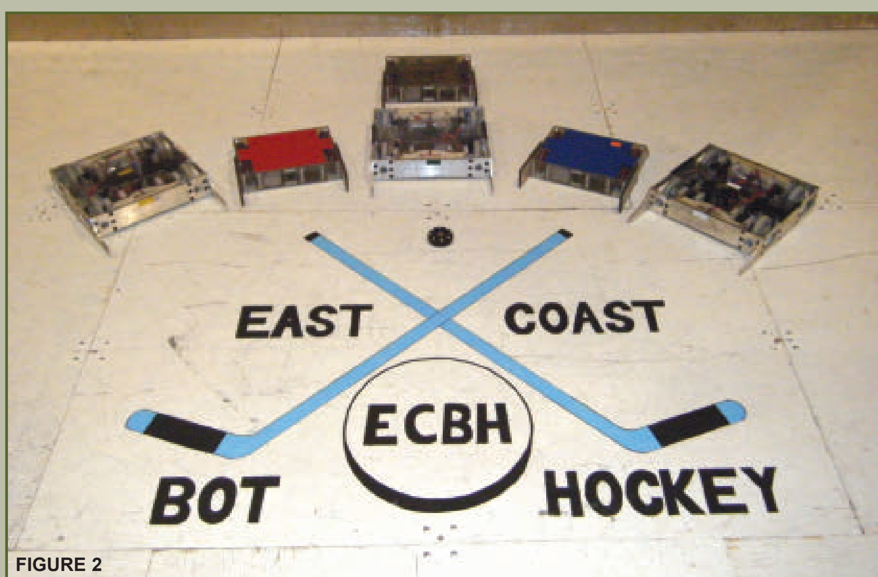


FIGURE 2

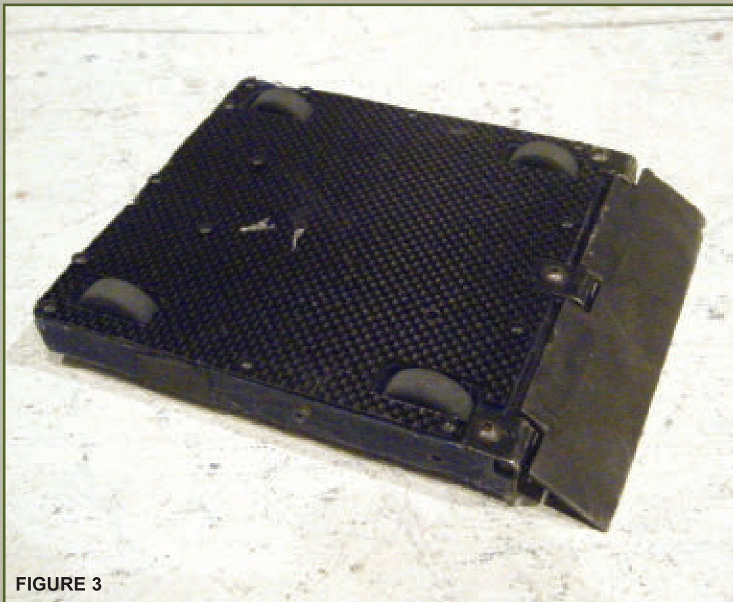


FIGURE 3

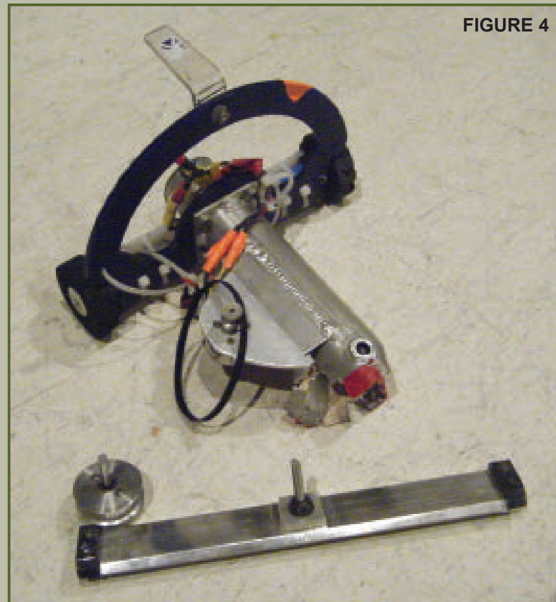


FIGURE 4

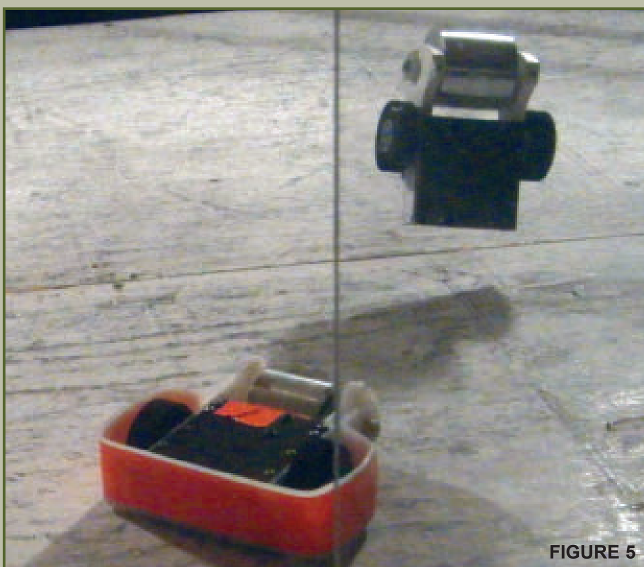


FIGURE 5

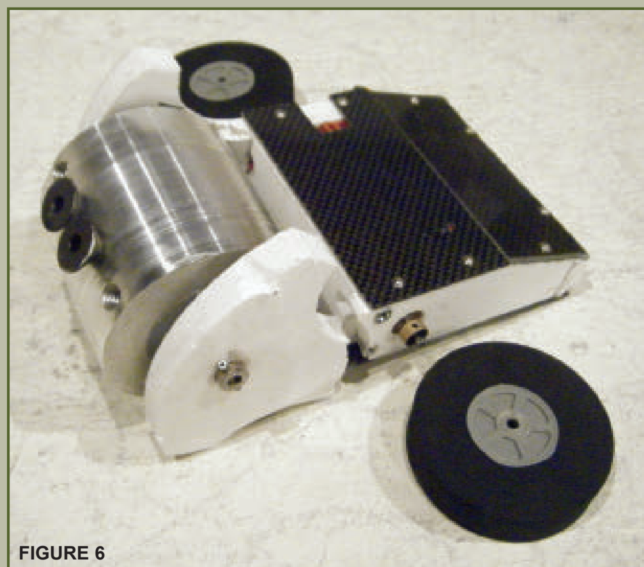


FIGURE 6

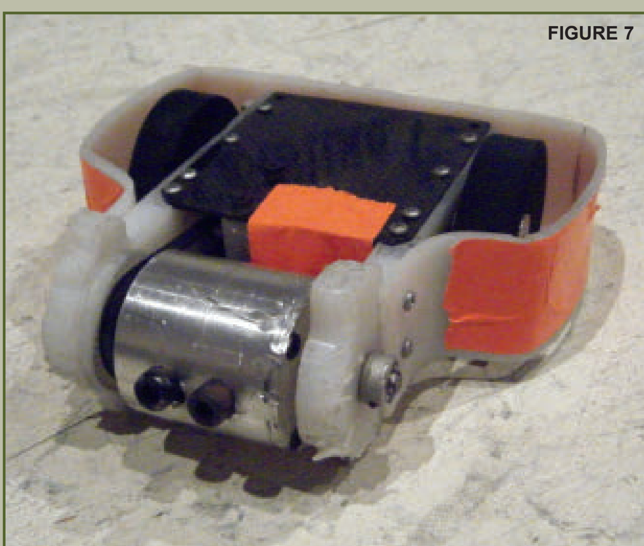


FIGURE 7


FIGURE 8

RESULTS

1 lb — Ants:
1st — Gilbert

3 lb — Beetles:
1st — Weta, God of Ugly Things

Bot Hockey:
1st — Team Pneusance

FIGURE 9

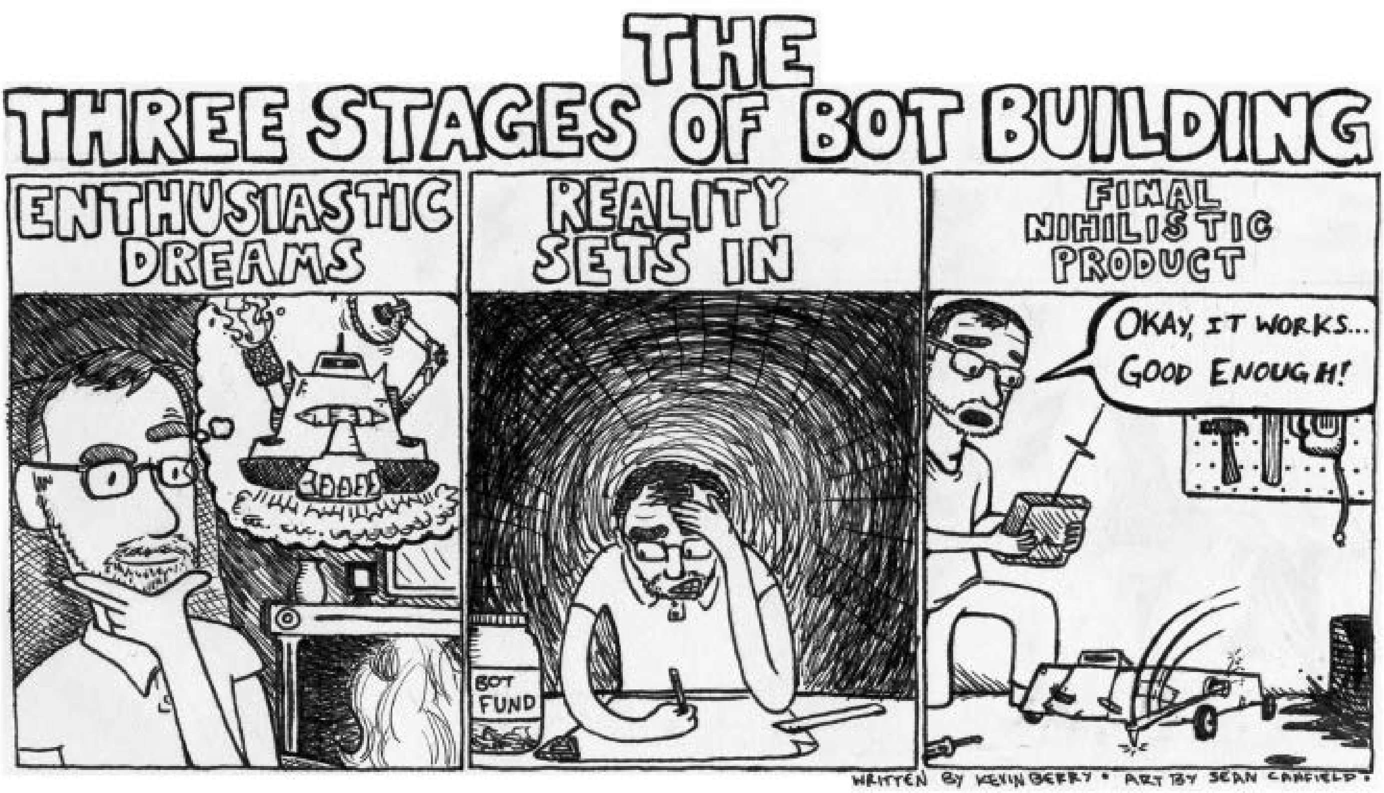
robots fought round robin. The Antweights were won by tough wedge Gilbert (**Figure 3**) after some hard fights with Poco Tambor, Uncle Bob, and The Tank.

The Beetleweights proved more destructive. Pure Dead Brilliant got wrecked (**Figure 4**) in a pyrrhic victory over MA2, and Grande Tambor got air time against Weta (**Figure 5**) before self-destructing against Cloud of Suspicion and ending up with a broken side wall and no wheels (**Figure 6**).

Weta, God of Ugly Things (**Figure 7**) escaped any serious damage and took first place.

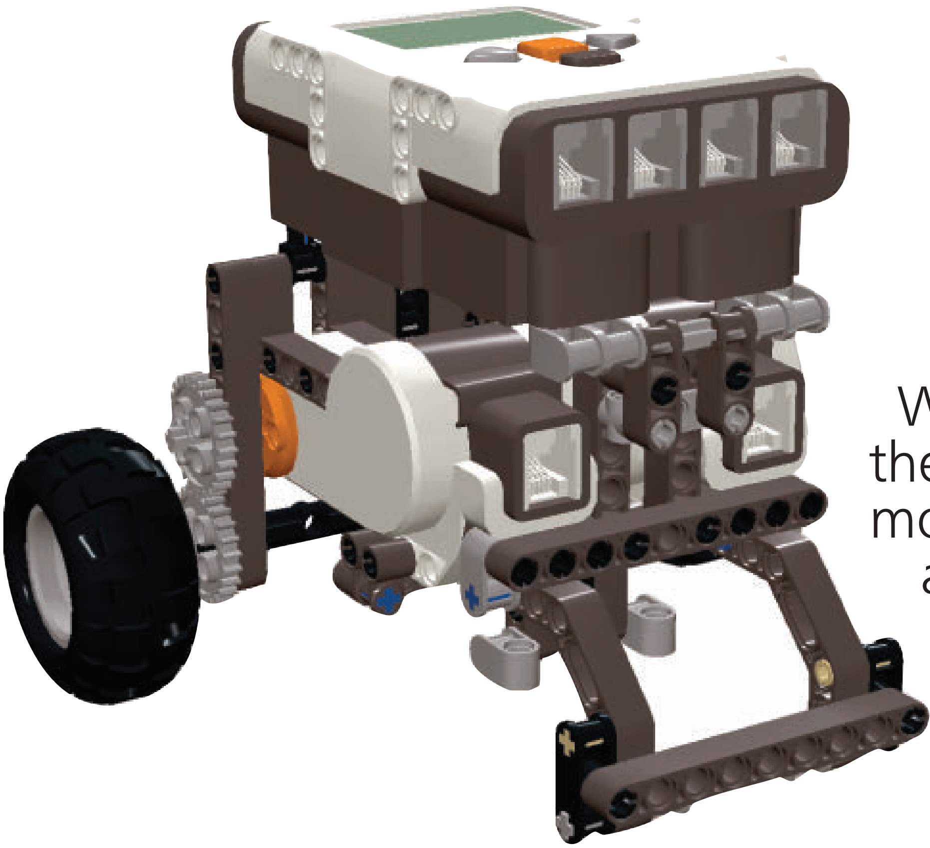
There were other robotic exhibits outside the competition hall, most notably a Bomb Disposal Robot (**Figure 8**) in the foyer and a feminine R2 unit prowling the cafeteria (**Figure 9**).

The Schiele Museum is hoping to have us back next year so this might be the start of an important regional event. The combination of good facilities and ready-made audience is hard to beat. **SV**



The NXT Big Thing #4

A Feast for the Sensors



We're busting out the sensors! Last month, we got acquainted with our good friend, the light sensor.

This time, we'll take it a step further, learning how to use two light sensors to accurately follow a line in any direction. Along the way, we'll learn about programming logic, and once we've finished our program, we'll wrap up and lead into next month's focus: using data wires to create dynamic variables!

Let's take a look at applications of multiple sensors:

- Why use multiple sensors?
- How can we determine what sensors to use?
- What are the best uses of dynamic variables?

Then, we'll get straight into programming!

Why use multiple sensors?

That's an easy one! For the same reason we [humans] have multiple senses. Adding a new sensor to a robot is literally giving it another sense. Sensors are extremely helpful in most cases — though it's important that you don't overload your robot with too many sensors. Otherwise, your programming will become overly complex.

How can we determine what sensors to use?

Consider what goal you're trying to accomplish when considering what sensors to use. Look at the environment your robot will be working in. Are there markings on the floor it could follow with a light sensor? Are there walls it could detect with a touch sensor? Is there a sound it could approach using a sound sensor?

If you're having trouble deciding on what sensors to use, try making a bulleted list of every element of the environment your robot is performing the challenge in. Your list might look something like:

- Walls
- Floor
- Challenge mat

Now make sub-bullets for each one, describing possible sensor uses. Like this:

- Walls
  - Ultrasonic sensor: Sense distance from walls
  - Touch sensor: Follow walls
- Floor
  - Light sensor: Detect whether on floor or challenge mat
- Challenge mat
  - Light sensor: Follow lines on challenge mat
  - Ultrasonic sensor: Detect nearby objects

The MINDSTORMS NXT kit has several variations, and within them, several software variations. This article is based on the NXT kit available from LEGO Education, rather than the retail version. Additionally, the software used in this article is the MINDSTORMS Education NXT Programming version 2.0. That said, if you have a different kit or different software, have no fear — chances are you'll be fine. Just know that you're on your own with technical issues.

Once you've made an exhaustive list of possible sensor uses, determine which sensor or sensors would be simplest and most efficient to use to complete the challenge at hand, and go for it!

What are the best uses of dynamic variables?

While not necessarily related to using multiple sensors, dynamic variables are another powerful tool in making complex programs with the NXT. A variable is a piece of information in your code that can change based on your needs. There are three types of variables in programming and their equivalents in the NXT software:

- Binary (Logic)
- Boolean (Number)
- String (Text)

Binary or logic variables are either true or false. One prevalent example of a binary variable is using a touch sensor — either it is pressed (True) or depressed (False). You can also use binary variables to make "conditional statements" like "If the variable is true, continue the loop; if the variable is false, terminate the program."

Boolean or number variables are numbers. When you tell your light sensor what value of light to look for, you're using a boolean variable. They're primarily used for storing values from sensors — as well as doing math — and storing the results. For this reason, boolean variables are commonly used in the NXT software to convert sensor values into numbers for things like steering. For instance, if we wanted to make a robot with a sound sensor that would go in a straight line if it got a reading of 50(%), turn hard left at 0, and turn hard right at 100, we'd need to create a formula to translate those numbers into 0, -100, and 100 respectively, and use a "data wire" to connect the output to

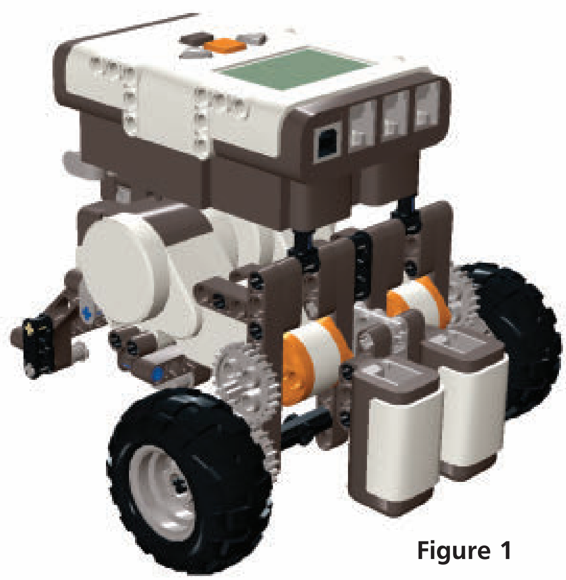


Figure 1

Figure 2

to the ground, thus giving us more accurate readings. Plug the left sensor into port 1, and the right sensor into port 2 (**Figure 2**). Now let's talk programming logic.

Thinking Like a Robot

So, we've got Eddie set up with two light sensors. Now we just need to understand what to do with them.

- If your floor is light in color, find some dark tape. If it's dark, find some light tape.
- Lay the tape out on the floor in a zig-zag pattern.

With two light sensors, we have several options for how we can follow the line:

- We can put one light sensor on either side of the line, and have Eddie turn into the line when one of the sensors detects it.
- Have one sensor following the line, and one sensor following the floor. If either sensor senses the opposite, the robot turns appropriately.
- Have both sensors on the line. If the left sensor detects the floor, the robot veers slightly to the right, or vice versa.

In this article, we'll be using the first of the listed options for following a line. As you'll see, even though we've already narrowed down what we're programming the robot to do, there are many different ways to program the robot to ultimately do the same thing!

- Set Eddie on top of the tape, and make sure that the light sensors are far enough apart that when Eddie is centered on the tape, one light sensor

the "steering" tab of a move block. More on that later.

String or text variables are rarely used with the NXT since the only way to output text is on the small screen which usually isn't accessible when a robot is in motion. String variables store text. For instance, a string variable might be "Hello World!" They're rarely used, and when they are, it's often for simple chat robots which interact with a user through text. With all that in mind, let's get to the robots!

Revisiting the Line Follower

Last month, we made Eddie follow a line using one light sensor, and built an attachment to follow a line using two light sensors. This time, we'll put that attachment to work! Let's start by attaching it to the robot. For double light sensor attachment building instructions, see the October issue. **Figure 1** shows Eddie with the double light sensor assembly attached. Note that it is in the lower of the two potential positions. This means the light sensor will be closer

Figure 3

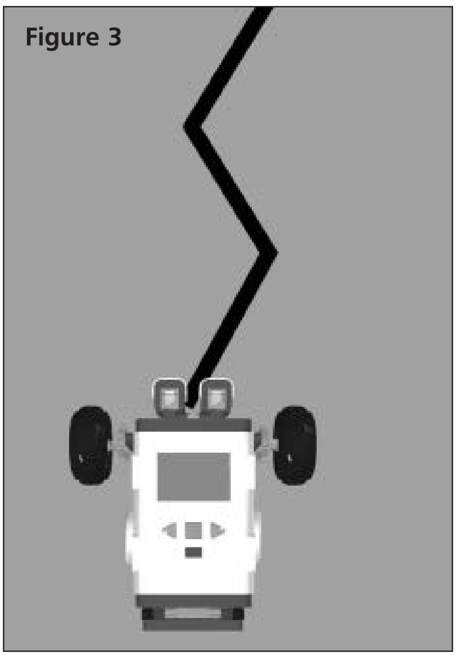


Figure 4

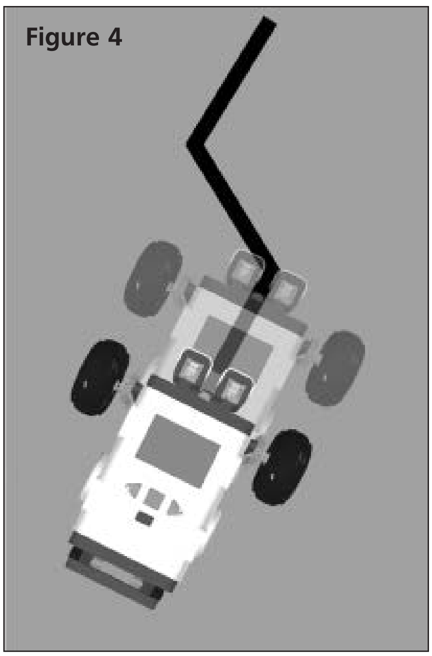
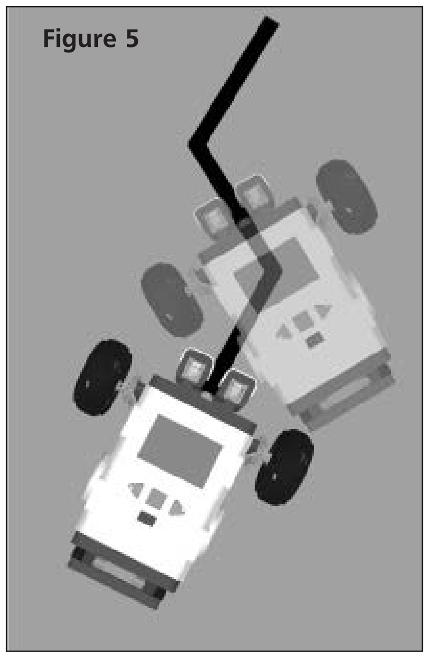


Figure 5



is on either side (**Figure 3**).

- If your tape is too wide, you can either cut it thinner, or widen the light sensor attachment to accommodate.
- Ideally, you should have a bit of wiggle room for your light sensors. They should be about 1.5-2 times as far apart as the line is wide.

Now let's get a feel for exactly what our program should look like. Remember, we want Eddie to start with one light sensor on either side of the line, and go straight forward until the line turns, at which point it should turn into the line. Let's take a closer look (**Figure 4**).

This figure shows the first step of our program: the robot moving forward until the first bend of the line. Take note of which sensor is tripped when the line bends left (**Figure 5**). If the left sensor (port 1) sees the line, we want the robot to turn to the left, as it does in this figure. If the right sensor (port 2) sees the line, we want Eddie to turn to the right. Make sense? As it turns out, a good way to write the program is to have Eddie turn ONLY until the applicable sensor no longer sees the line meaning that it's back on track. Let's get started on the program.

Coding the Double Light Sensor Line Follower

While there are many ways to structure this code, we'll take a look at one basic, linear structure that should be easy to understand. Let's get started.

Dual Light Sensor Program Instructions

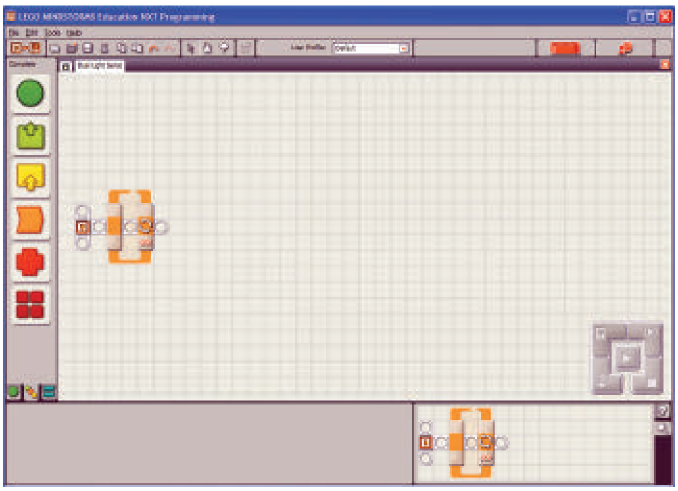


Figure 1. Start with an infinite loop, in which the rest of our program will take place.

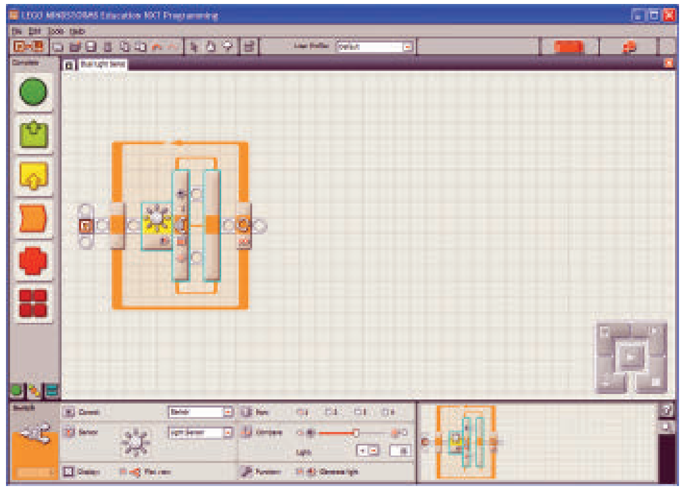


Figure 2. Add a "switch" controlled by a light sensor. Ensure that you select port 1, and find a threshold value appropriate for your lighting conditions.

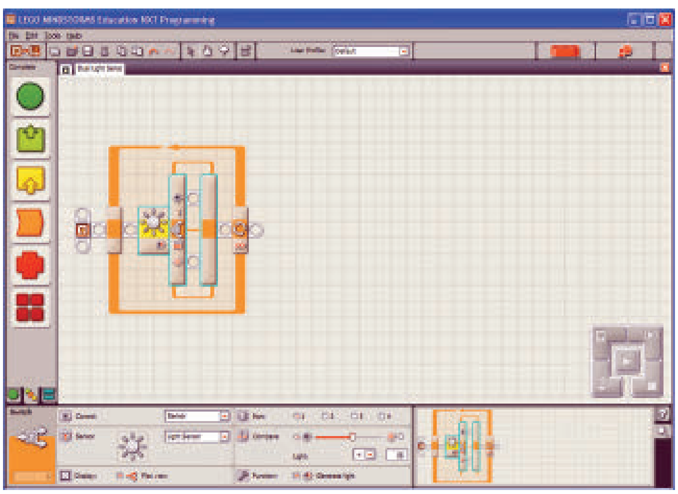


Figure 3. If Eddie sees the floor (lighter than the tape) with his left light sensor (port 1), then we want him to spin his motor backwards, making the left wheel roll forwards! (Be sure to set duration to "Unlimited")

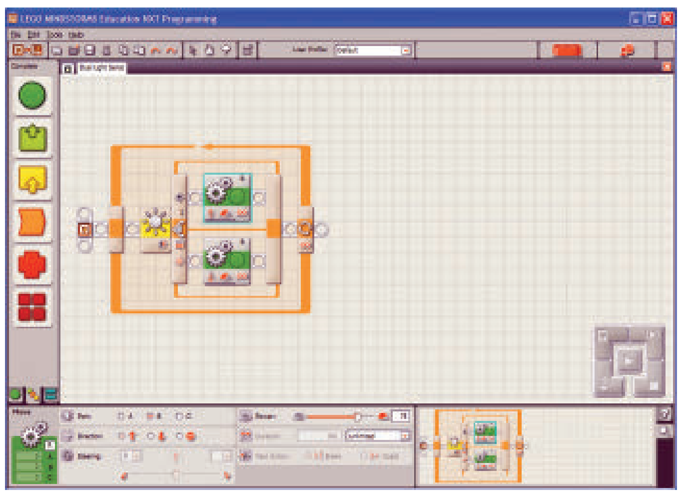


Figure 4. If Eddie sees the dark tape with his light sensor on port 1, we want him to spin his motor forwards, so the wheel rolls backwards! (Be sure to set duration to "Unlimited")

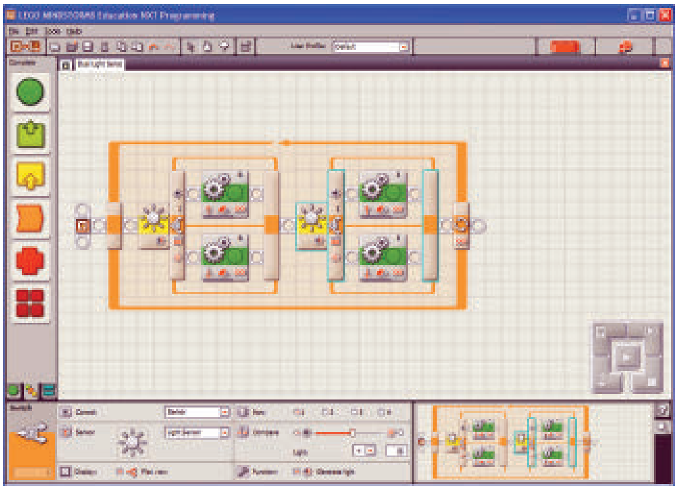


Figure 5. Now hold the control key and click the light sensor switch, dragging it- which will create a duplicate of the switch and everything inside it! Drag that duplicate into place just after the first switch inside the loop.

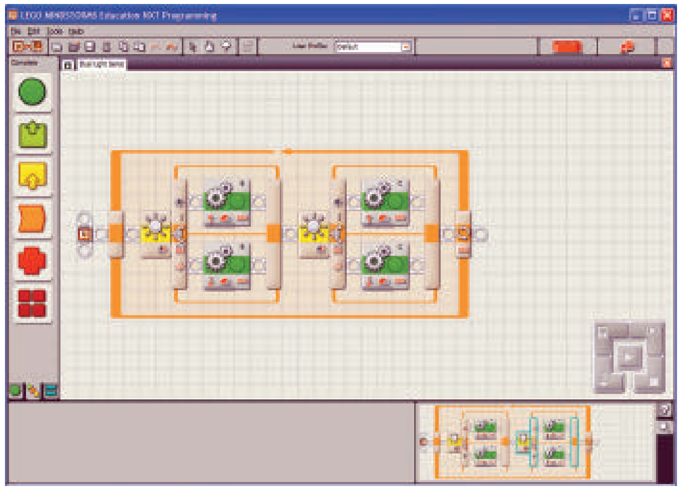


Figure 6. Finally, change the sensor port for the second switch to port 2 (the light sensor on the right of the robot) and switch the motor commands within the second switch to port C (the motor on the right of the robot).

If all goes well, you should now be able to download your program onto Eddie and watch him roll! In case you have any issues, here's some troubleshooting with common problems:

www.servomagazine.com/index.php?/magazine/article/november2010_Intermaggio

- Eddie ignores the line completely.
  - Check your light values. Make sure that you're waiting for a value between that of the floor and the line.
  - Consider lowering the motor power levels. Sometimes if your line is thin, he might be

moving so fast he misses it!
- Eddie goes backwards when I told him to go forwards.
  - Remember that you have a gear train. This means that telling Eddie's motor to spin forward results in him rolling backwards, and vice versa!
- Eddie turns the wrong direction when he reaches the line.
  - First check to make sure that your left light sensor is connected to port 1, and your right sensor is connected to port 2.
  - If that doesn't work, check your programming. You want to associate sensor 1 with motor B and sensor 2 with motor C.

Experiment!

Now that you understand the basics, do some experimentation! Here are some ideas:

- Find another way to write a program with the exact same output — Eddie follows the line with one sensor on either side.
- Write a program to follow the line with one sensor on the line and one sensor off.
- Try setting the light sensors further apart, then closer together.
- Try making Eddie follow the line backwards.

Wrapping Up

This time, we programmed Eddie to follow a line using multiple light sensors which makes for much smoother, more accurate line following than using just one sensor. We also learned a bit about programming logic, and using switches to check whether something has happened (in this case, checking to see whether Eddie is on or off the line!).

Next time, we'll be doing something entirely different with the light sensors. Using two light sensors facing-up, we'll be writing a program that uses dynamic variables to drive Eddie towards the brightest [or darkest] point in a room. Stay tuned! **SV**

GoPHR

An Inexpensive Prototype Platform for General-Purpose Household Robotics or Telepresence

By Alan Federman

Like most people my age, I grew up watching the Jetson's on TV (back when portable TVs took only two guys to move!). I have given up on the flying car, but I at least expected a Rosie by now. When I got my first 'personal' computer back in 1980, my sister wrote me a letter that said "Wow, personal computers, can personal robots be far behind?" Well, it has been 30 years and it's about time. Since they aren't available at a reasonable cost just yet, we'll have to build our own. I'm going to show you how I did it.

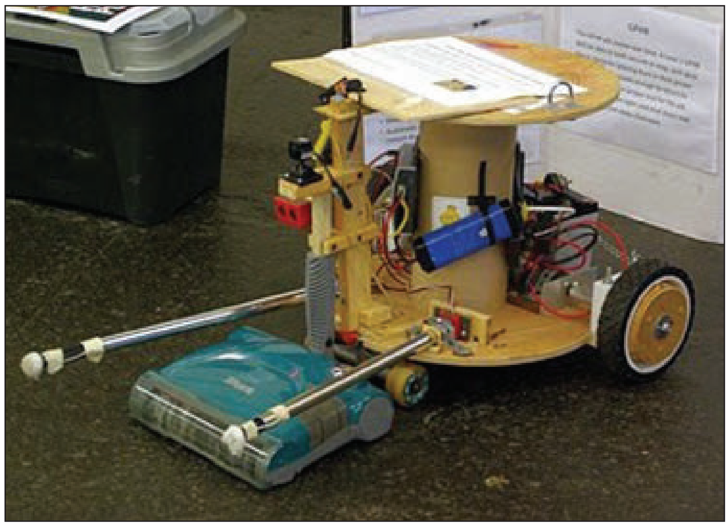


FIGURE 1. GoPHR mark2 shown as a telerobot with various attachments.

www.servomagazine.com/index.php?/magazine/article/november2010_Federman

Seriously, this project came about after I agreed to be a substitute speaker at a meeting of the Homebrew Robotics Club two years ago. The talk was about building practical robotics platforms. The problem, as I see it, is that building a platform that can do something useful is a major hurdle for the beginner. While platforms like the Boebot, iRobot Create, and LEGO NXT are great entry level introductions in my opinion, they are too underpowered to perform the simplest household task; for instance, move a 'Shark' type cordless sweeper across the room. I also have a problem with buying one robot to sweep the floor and a second robot to mop it since 95% of the robot platform would be the same. I'd like something that would be easy to repurpose — something like the 'snap-in' tools concept. This led to the GoPHR concept.

GoPHR stands for General-Purpose Household Robot. The idea is to develop an open source hardware platform capable of performing useful household tasks and be expandable in both the hardware, controllers, and software. Just like DARPA pushed the art of robotic cars, the art of domestic robotics could be spurred by staging challenges.

- Level 1: Sweep the floor, return to charging station.
- Level 2: Pick up the smelly sneaker, return it to the proper place, then sweep the floor.
- Level 3: Be able to open cabinets and doors while navigating.
- Level 4: Be robust enough to venture outside and fetch the newspaper.

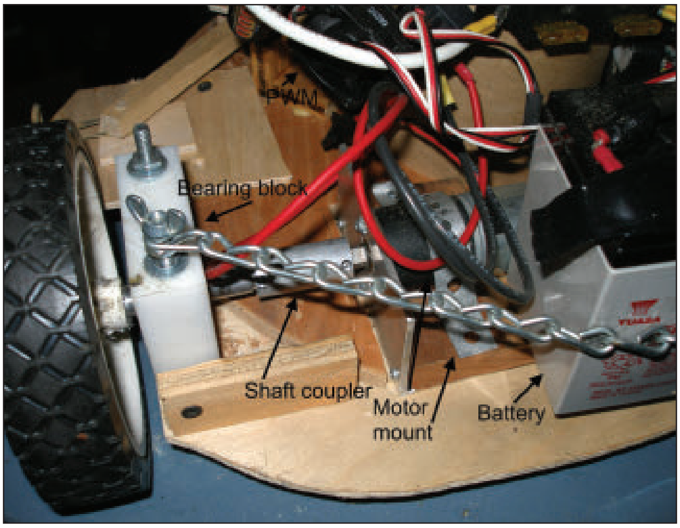


FIGURE 2. Motor mount and battery detail.

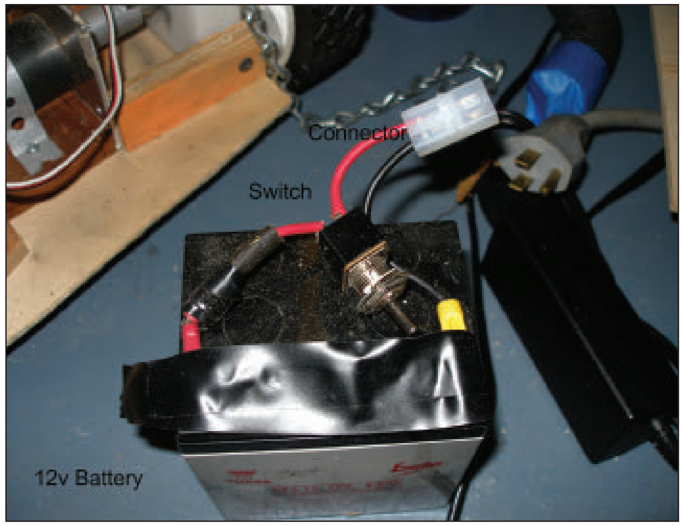


FIGURE 3. Battery wiring.

After developing the platform up to Level 1, I saw it would be easy to convert it between fully autonomous, fully telerobotic, or somewhere in between. This platform can accomplish much of what platforms like the Anybot or Texai can at a fraction of their costs.

Construction Details

The original GoPHR platform was built in just a day. I was going to use a plywood (excuse me, naturally derived engineered layered composite) sheet, but I happened to find an empty cable spool that was the perfect size. This saved me from trying to find a band saw or using a hand jigsaw to make the curved cuts. Since the initial design worked so well, I acquired a second cable spool, and used the plywood disks to add two more 'decks' to the robot. I used steel electrical wire conduit to attach the decks together. **Figure 1** shows the entire GoPHR set up as a telerobot.

Current attachments are a sweeper ("Shark") and a damp mop (Swiffer). A spatula/scooper will be added to a deployable arm on the mid-deck in the near future.

The motors are two 12V Globe motors. These use planetary gear reductions similar to what is used in cordless drills. As a result, they are somewhat noisy. An alternative would be to use a motor with a built-in worm gear reduction. These are usually quieter, but may be harder to mount. The motors were attached to the chassis by constructing mounts from aluminum angle, scrap wood, and plumber's metal tape. These were attached to lawn mower wheels using an aluminum axle and an axle bearing block made out of plastic. (This was surplus cutting board obtained at nominal cost from a plastics wholesaler.) The most challenging machining issue was fashioning the shaft couplers that join the motor to the wheel axle. This was done on a metal cutting lathe using 3/4" aluminum rod, and then drilling and tapping #10 machine screws to attach the coupler. (See **Figure 2**.)

I used a 12 volt power system since automotive parts are readily available, as are the sealed lead-acid batteries used in UPS backups. It helps to use good connectors and switches; be careful to tape all bare battery cables. I am sorry to say I learned this lesson the hard way when the chain holding the battery in place slipped and shorted

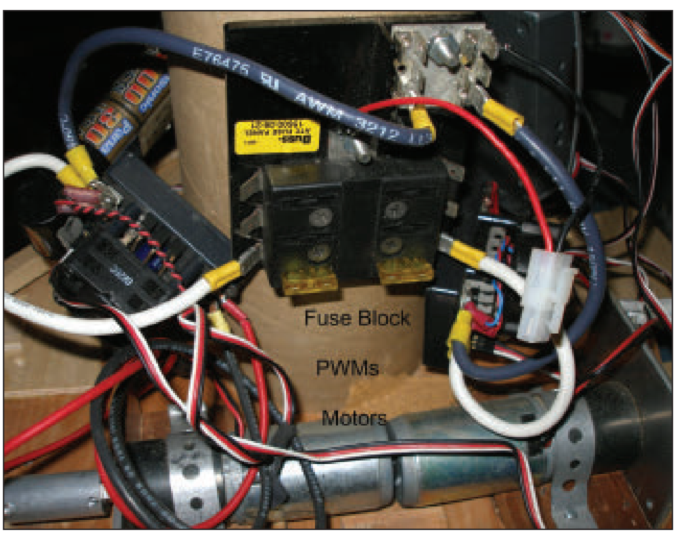


FIGURE 4. Motor power and control.

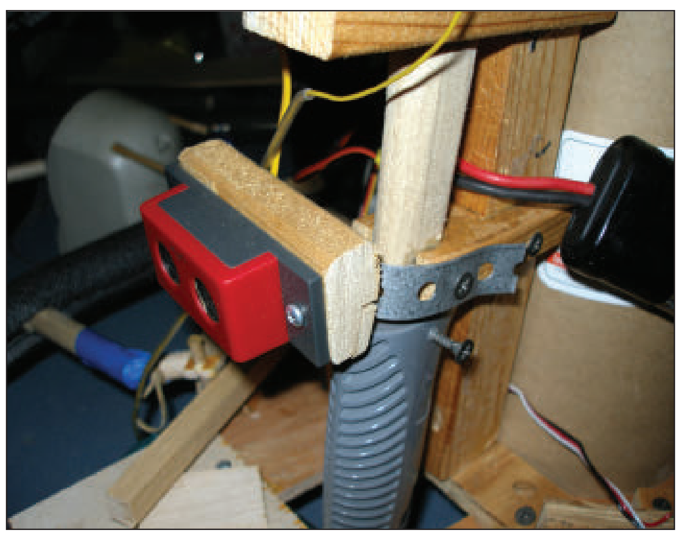
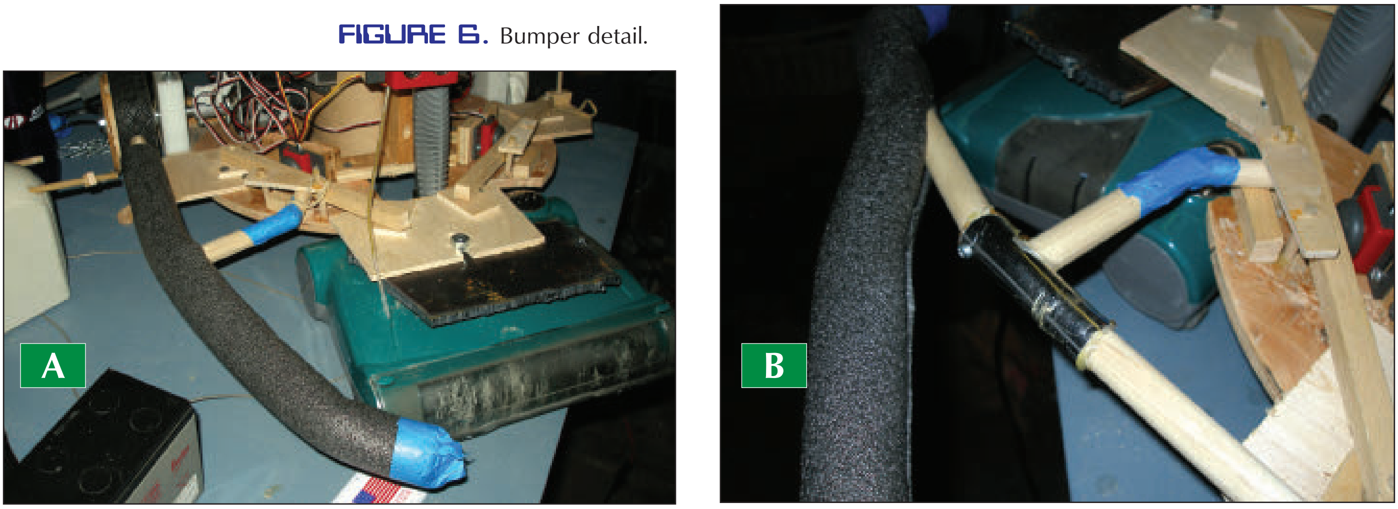


FIGURE 5. Ultrasonic range finder.

FIGURE 6. Bumper detail.



across the terminals. Fortunately, only my pride was hurt. **Figure 3** shows the battery wiring.

While my controller would run happily off the battery, I decided to isolate the controller power from the motor power system to avoid spikes and noise. Automotive motors can generate voltage spikes and RF noise that is potentially harmful or disruptive to more sensitive electronics. I used a fuse block and PWM (pulse width modulation) units to control the power to the motors. Take a look at **Figure 4**.

Figure 4 shows the wiring of the fuse box and PWM units. The wiring scheme is red for V+, black for ground, and white for the fuse protected 12V. The three wires (black/red/white) are for the control signals to the PWM units. For this robot, I used older IFI Victor PWM units but less expensive PWM units or even relays would suffice for this application. Outside of the controller, the most expensive components were the motors and motor controllers. I used two kinds of sensors for this robot: an ultrasonic range finder (**Figure 5**) and two bump sensors. Getting the bumpers to work correctly took a couple of tries. I finally came up with the design shown in **Figure 6A** and **6B**. I used some 1" wood dowel, some aluminum tubing, and pipe insulation to fashion the bumpers. The sweeper I use is pretty light in weight, so some additional ballast is needed so it doesn't tip over, especially when adding weight to the top. I'm starting to add a remote telepresence functionality to GoPHR, as well.

So far, my software development has been limited. I have been using an older VEX controller. I was hoping to get my greasy grubby hands on one of the new Qwerk type controllers being developed in stealth mode. Alternatively, a laptop plus a Serializer board would have the necessary processing power to run as a telerobot. It would be nice to run this robot as a remote website (a movable webcam). That way, I can get the house vacuumed in case I am running late. I could call up from my hands-free mobile and tell GoPHR to get GoING. Alternatively, I can just ditch the sweeper and find out what the cat has been up to.

I am beginning to work on a manipulator for simple tasks. I plan to evolve this manipulator by adding a degree of freedom at a time. My first goal is to add a spatula, because then I can have a robot that can both sweep the floor and flip burgers (which appear to be the career choices I have left during this down economy). Another planned enhancement is a docking charging station. Depending on my choice of motors, materials, and PWM units, the cost of a GoPHR platform (sans controller) will be in the $200-$500 range.

For additional information and updates, check out my website at **http://federman.best.vwh.net**. SV

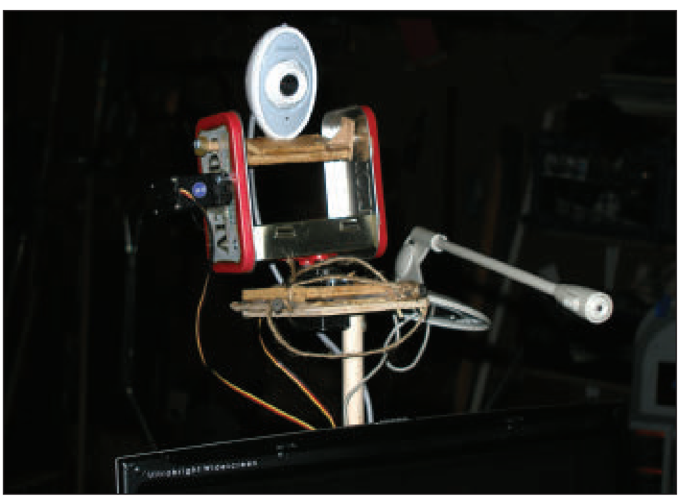


FIGURE 7A. Pan tilt webcam (re-tasked Altoid boxes) with microphone.

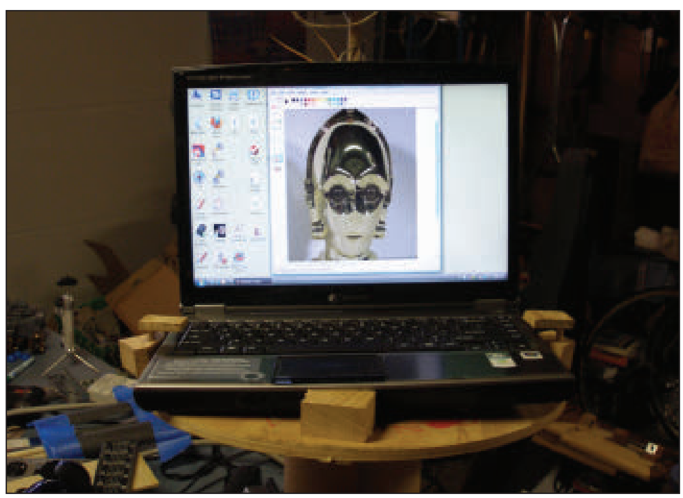
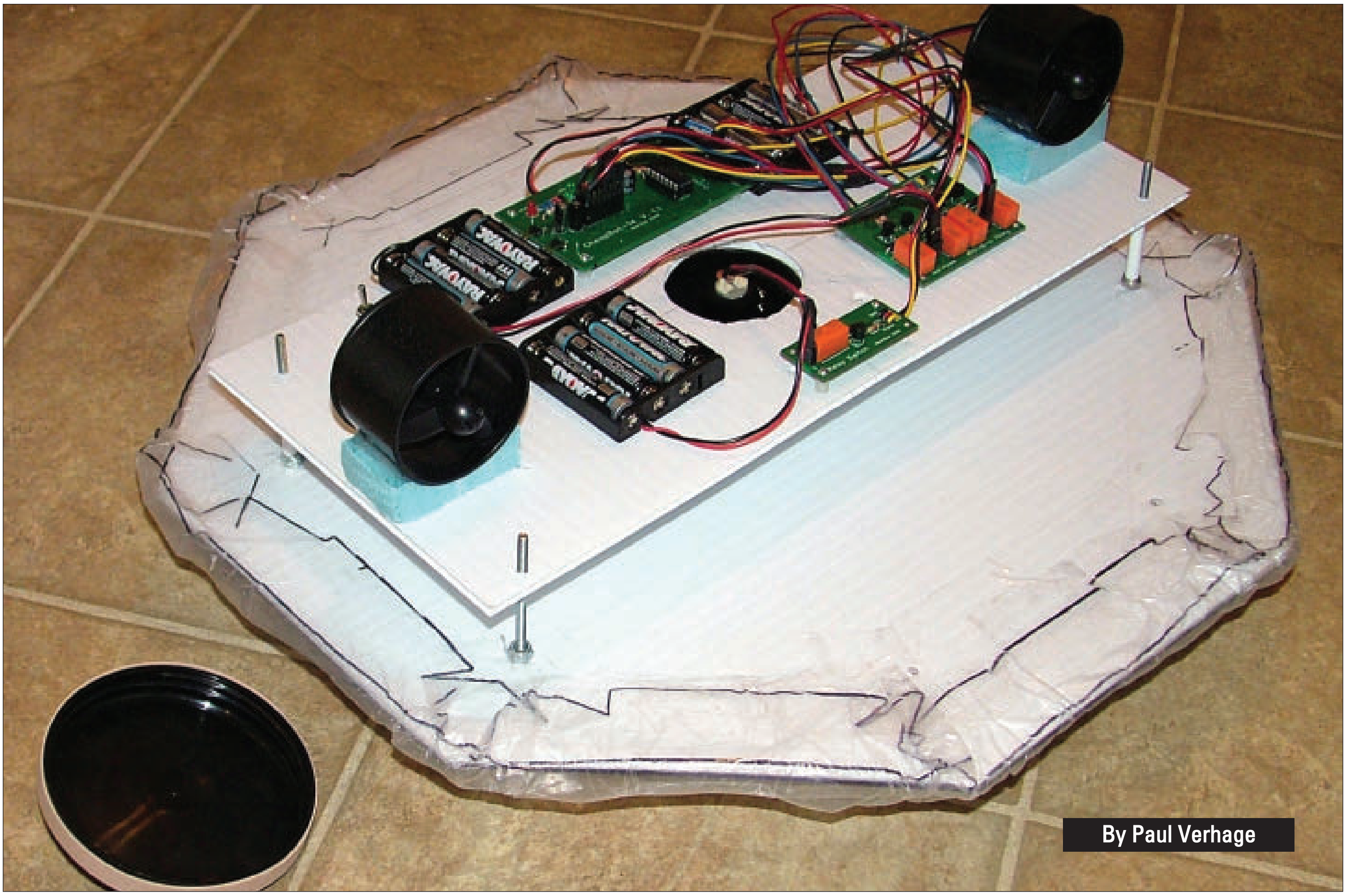


FIGURE 7B. A Skype call to 3CPO.


RobotShop
Robotics at your service! ™
Robots, lots of robots!
The World's Leading Source
for Domestic and Professional Robot Technology
www.robotshop.com

HOVERBOT:

You Picked A Fine Time To Leave Me, Loose Wheel!



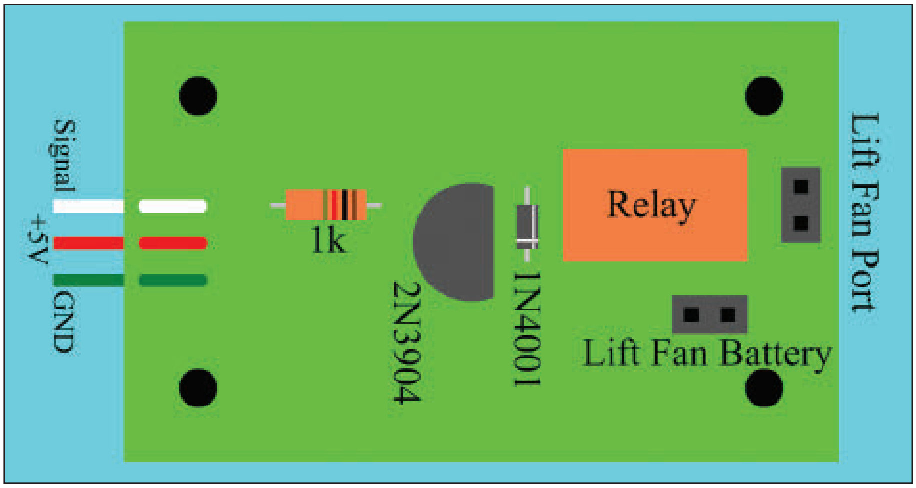
By Paul Verhage

We've covered building the body of a floating robot and its drive mechanism. So this month, we'll wrap up the HoverBot with an explanation of its drive electronics. After that, you'll have all the information you need to build your own HoverBot. How you ultimately control your HoverBot I will leave up to you.

The ducted fans of the HoverBot spin quite fast. So, rather than risk damaging their motors by slamming them from clockwise to counter-clockwise, I opted for a slower approach to controlling the fans: relays. The fans draw nearly two amps of current and using a traditional transistor switch would mean losing 0.7 volts of the fan's voltage and dumping 1.4 watts of heat through the transistor. (I'm also trying to keep the HoverBot light and don't want to add an additional AAA cell to make up for the voltage drop.) Therefore, the HoverBot uses DPDT relays instead of transistors or MOSFETs. The relays have a must-latch voltage of five volts but a coil resistance lower than I like (they require the robot controller to source more current than I'm comfortable with). To get around the high current requirements of the relays, I used a 2N3904 transistor to source the current for each relay. The current required to operate the transistor is so small there is very little wasted power (about 210 mW) switching a relay on and off with it. The HoverBot uses five relays in two circuits to control the three ducted fans in the HoverBot.

Lift Fan Circuit

The first circuit is the one that controls the main lifting



This shows the placement of parts for the lift fan circuit. The arrangement of wires in the circuit's control cable is the same as a servo, so I recommend using a similar color scheme.

fan. This circuit gives the HoverBot the ability to take off and land under automatic control (pretty cool). It only takes the HoverBot a second to fill its skirt and lift off, or to dump its skirt and land. To reduce the number of different parts, the HoverBot uses the same relay for both the lift fan and the drive fan circuits. The fact that the HoverBot only needs a SPST relay to control the lift fan means the circuit wastes half the relay. While probably not necessary, a reverse-biased 1N4001 diode across the relay coil protects the robot controller from the relay coil's kickback.

The three holes at the bottom of the copper pattern (available in the *SERVO* downloads) are strain relief for the wires connecting the circuit to the robot controller. The four larger pads in the corners of the PCB are its mounting holes. Use small nylon spacers or a foam neoprene sheet and 2-56 bolts to mount the lift fan circuit to the HoverBot's top deck. I strongly encourage the use of nylocks because like an airplane, you don't want FOD (foreign object debris) getting into the HoverBot's fans. The operation of the fans will loosen up regular nuts over time.

The HoverBot relays are NEC DPDT relays; they are available from Jameco as part number 2081650 and cost $2.49 each. You'll need to order five of them to build the two circuit boards for the HoverBot. This is a list of the remaining parts you'll need to make both circuits for the HoverBot.

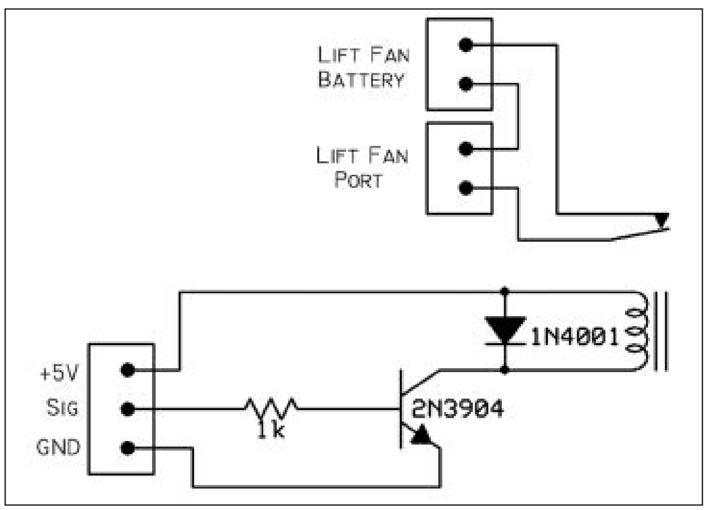
- Five 1N4001 diodes
- Five 2N3904 NPN transistors
- Five 1/4W 1K ohm resistors
- Wire (#24 gauge stranded)

I also recommend getting a single row receptacle that's 11 or more pins wide. Using receptacles allows you to plug the components of the HoverBot together, rather than soldering everything together. Note that there are two ports in the lift fan circuit. The first — the ducted fan port — is where the ducted fan plugs into the circuit. The other port — the battery port — is where the four AAA battery pack plugs into the circuit. Note that if after you plug everything together and the HoverBot sucks itself to the ground at take-off, you plugged the battery pack in backwards (no harm done, just switch it around).

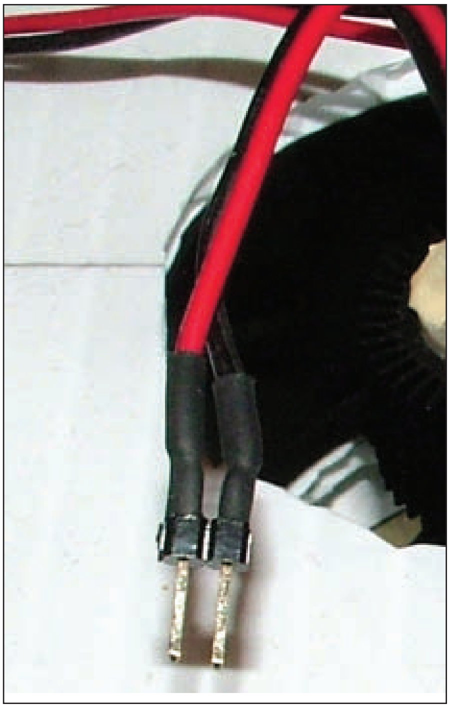
You'll find the length of the ducted fan wires will need to be extended to reach the circuit boards. GWS does not terminate the wires of the ducted fan, so you'll need to solder them to a two-pin header and cover the soldered connection in heat shrink for protection. Now you can plug the ducted fans into their appropriate receptacles.

Drive Fan Circuit

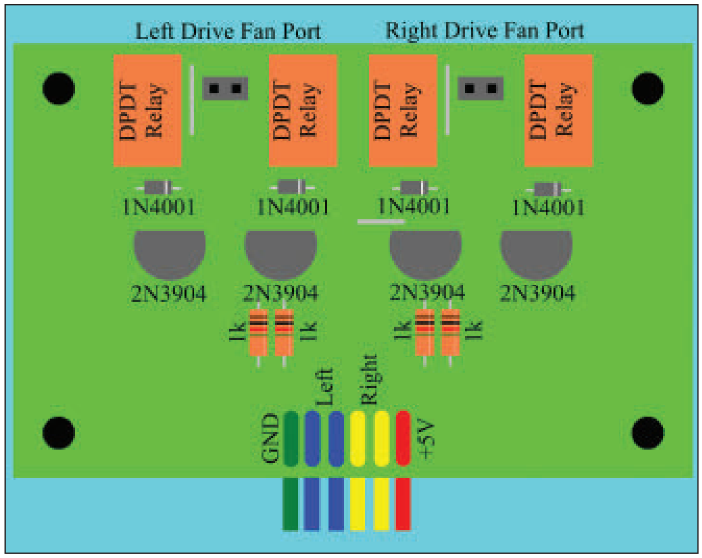
Unlike the lift fan circuit, the drive fan circuit uses both sides of its DPDT relays. Like other H-bridges, it requires two input pins to operate each drive fan. Two input pins permit four combinations of outputs. When both input pins are set



Just a typical relay circuit. The robot controller saturates the base of the 2N3904 with 4.3 mA of current so the transistor will let current flow to the relay coil. The closed contacts inside the relay then route current to the lift fan without a voltage drop to the fan. Note that the schematic only shows the normally opened side of the DPDT relay; the normally closed side is not used.

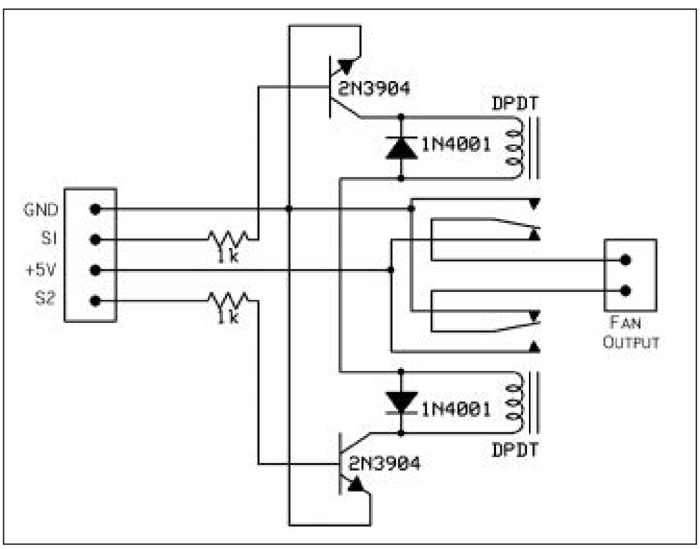


The leads of this ducted fan were soldered to the short side of a two-pin header. I slid heat shrink over the wires after tinning them, but before soldering them. Insert the pins into a receptacle before soldering or heat shrinking them. That keeps the pins in the proper alignment while they are getting hot – and possibly softening their plastic jacket.



I recommend color coding the wires in the circuit's cable system similar to what is illustrated here. The drive fan circuit requires three jumper wires. Use cut resistor leads for the jumpers. Be careful the transistors and diodes are not soldered backwards.

the same (either both HIGH or both LOW), the voltage across the fan output is the same so there is no voltage difference to drive the fan's motor. Setting the input pins opposite to each other (one HIGH and the other LOW) creates a voltage difference that appears across the fan output and the fan motor spins either clockwise or counter-clockwise. There is no braking function in this H-bridge; the ducted fans only slow down due to friction. I figure that's more like a rocket thruster anyway.



This is the schematic for one of the drive fan's H-bridges (this circuit is duplicated on the PCB for the other drive fan).

Put It All Together

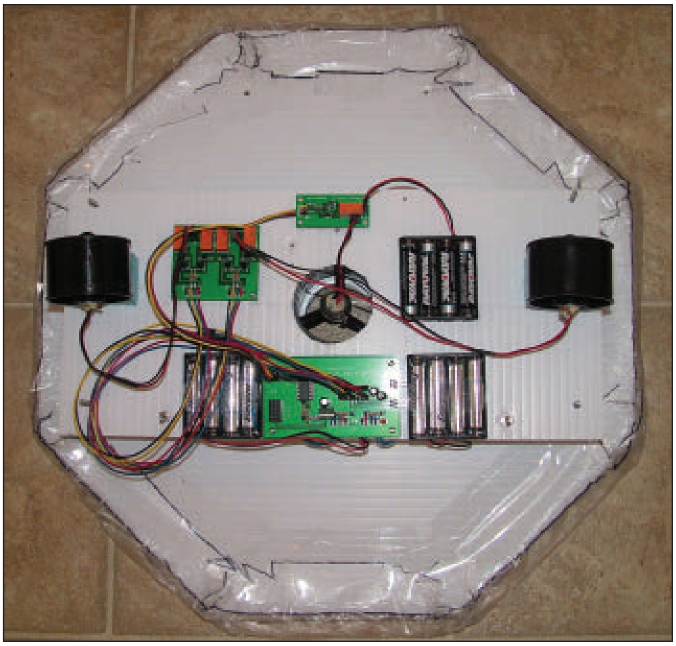
Bolt all the HoverBot's electronics, the lift fan circuit, drive fan circuit, robot controller, and the battery packs to the top deck. Do not bolt them to the bottom deck as it breaks the airtight seal of the skirt and makes it difficult to move the electronics around at a later time. Position the electronics on the top deck so it distributes their weight as evenly as possible. I also suggest placing as much stuff as close to the center of the deck as possible.

I've experimented with using lithium cells for power. They're much lighter than alkalines and as I understand it, they are used in electric airplanes. However, I'm still not convinced they can provide the high current the HoverBot's fans need. So, use battery packs for power and you can swap out the battery types to determine for yourself which battery chemistry works best.

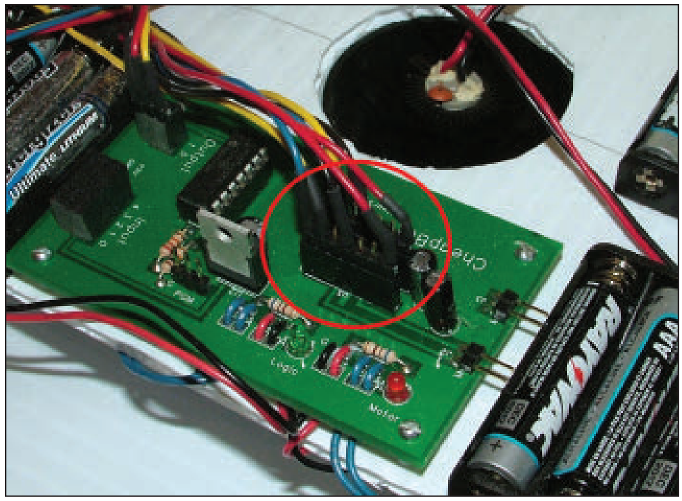
I use the same subroutines to drive and turn the HoverBot that I use to drive and turn a wheeled robot. The subroutine to drive a ducted fan forward looks like this:

```
Forwards:
   high 2
   low 3
   low 4
   high 5
return
```

However, the code has to use reverse thrusters every



I tried to keep the heavier battery packs near the center of the top deck of my HoverBot, but didn't give it enough thought prior to bolting everything together. The robot controller is one of my CheapBot-14s.



I circled where I replaced the Toshiba H-bridges I normally use on my CheapBot robot controllers with the cable to the drive fan H-bridge. The receptacle in the robot controller made it easy to swap out the Toshiba H-bridges for the NearSys Relay H-bridge.

time it needs to come to a stop (translating or rotating). To drive the HoverBot forward, arrest its forward motion, turn left, and then arrest its turn. I use these lines of code:

```
HoverBot:
  gosub Forwards
  pause 2500
  gosub Backwards
  pause 2000
  gosub Left
  pause 800
  gosub Right
  pause 700
  gosub Freeze
```

Sounds simple, right? Well, my HoverBot has a tendency to rotate, even when it's programmed to only translate. I have come up with three possible sources of this anomalous rotation: asymmetry in the skirt; variations in the thrust of the drive fans; and counter-torque from the lift fan. Perhaps asymmetry in the skirt somehow creates unbalanced jets from the HoverBot's lift nozzle. If so, then it's going to be important to tape the plastic sheet to the bottom deck with as few wrinkles (or at least evenly spaced wrinkles) as possible. Once taped to the bottom deck, the skirt is difficult to remove without damaging it. Therefore, it pays to spend some time

attaching the skirt. I don't know how similar the fans are, however, I wouldn't be surprised if there is a small variation in the thrust between them. That small variation in thrust in a nearly frictionless HoverBot can certainly produce a rotation. Until I can find a way to test the thrust of the fans, there's no way to know how closely the fans match each other. Since I don't believe it's realistic to use a rheostat to adjust the spin of a ducted fan, asymmetry produced by mismatched fans may be something we have to work around. I suspect a small source of the HoverBot's spin is due to the counter-torque of the spinning lift fan blade. The ducted fan blade spins fast, but has little mass. The greater mass comes from the spinning armature inside the fans motor, but its radius is even closer to the center of rotation. Combined, the torque from the spinning ducted fan should be small compared to the rest of the HoverBot. However, the HoverBot is floating on a nearly frictionless cushion of air and even a tiny torque can add up over time. This potential (weaker?) source of anomalous spin is another one we'll have to work around.

Until I get a better handle on this anomalous spin, I'll have my HoverBot correct for it by dead reckoning. At times, the HoverBot will spin in the opposite direction for a moment. Perhaps later, I can try adding a small accelerometer or gyro to the HoverBot. An accelerometer lets the HoverBot detect the spin and as the rotation builds up, the HoverBot can produce a counter-spin to put itself back on track. Some time next year, I'll have the HoverBot position itself based on a model star tracker. Next, will come the larger, HoverBot 2.0. I'll keep everyone updated about my experiments via my tweets, my blog, and my YouTube Channel (@NearSys, **nearsys.blogspot.com**, and **www.youtube.com/nearsys**).

I hope you've enjoyed my HoverBot series. Again, feel free to assemble your own and keep me informed about your successes.

Here's to a nearly frictionless future without wheels. **SV**

By the time this article shows up in print, I should be selling HoverBot kits for those who don't want to make their own printed circuit boards. You'll find information on my webpage at **NearSys.com**.

Rate the eM8

By Fred Eady

Schematic diagrams are nice to have as they can convey the inner workings of a design with just a glance. When it comes to totally understanding an alien project, source code and a schematic are the ideal combination. A schematic diagram without source code is useless if you are attempting to port or revive the look and feel of the original application. However, it is a no-brainer to recompile the source code and reload the target hardware without the help of a schematic. What if you were presented with a piece of hardware, with no accompanying schematic and source code, written totally in assembler? Could you port the source code to C and retain the original look and feel of the assembler-based application?

The Hardware

The eM8 (pronounced eMate) hardware captured in **Photo 1** comes from the land down under and was created by Sam Wallace and Duncan Campbell as a teaching tool for students attending the Queensland University of Technology which is located in Brisbane, Australia. Commercial use of the eM8 is discouraged as the design contains digital and analog circuitry that is solely intended to illustrate basic microcontroller concepts in a classroom environment.

A Microchip PIC16F882 is the center of attraction, and costars with a Microchip MCP2200 USB-to-Serial bridge and a quartet of Kingbright ACSC04-41SURKWA-F01 Hyper Red seven-segment display modules. If you're new to the MCP2200, we discussed the design details behind it in the July '10 Design Cycle column over in *Nuts & Volts*. So far, by simply attaching a USB cable between the eM8 and my laptop, I've determined that the MCP2200 portal does not use the USB interface as a source of power. That means the MCP2200 is most likely used as a simple serial communications portal.

The four Kingbright seven-segment displays are driven in the traditional manner using an MC14511B BCD-to-seven-segment latch/decoder/driver. The ACSC04-41SURKWA-F01 LED display is a common cathode device. So, while the MC14511B is driving the LED anode segments, an MC14042B quad transparent latch seems to be switching four 2N7002 MOSFETs attached to the display module cathodes. You can make out the anode drive lines just above the MC14511B and the 2N7002 cathode drivers below the display modules in **Photo 2**.

If you're wondering how I know all of this about the eM8 LED display hardware, there's no rocket science behind the methodology. I simply read the manufacturer's markings on the SOT-23 devices and the part numbers on the ICs and displays. The good news is that I don't have to be right on the money as to what is what electronically. I can ride around on my donkey all day as the source code will provide the proper driver algorithms for the seven-segment display bank regardless of how I think the LED driver electronics are wired.

A MAXIM/Dallas DS1307 I^2C real time clock is also part of the eM8 design. Pad areas for three I^2C EEPROMs are available, but no IC sockets or EEPROM devices are installed on the out-of-the-box version of the eM8.

The eM8 documentation suggests that the electronics be powered with a quality wall wart producing between

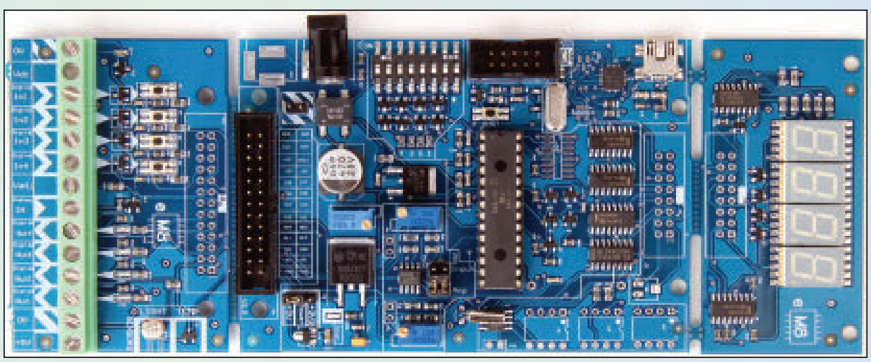


PHOTO 1. The eM8 looks really busy in this shot. It's not as daunting if you divide the electronic functions at the printed circuit board break-away points. From left to right, Photo 1 reveals an I/O section, a main computing and power section, and an LED display section.

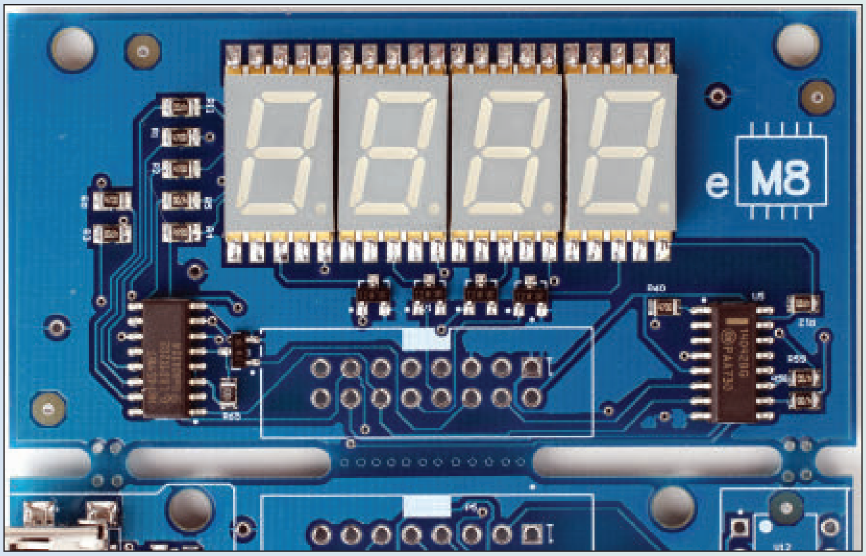


PHOTO 2. A set of ACSC04-41SURKWA-F01, 4511, 4042, and 2N7002 datasheets along with an ohmmeter is all we need to figure out how this industry-standard set of electronic components is wired together.

under the lens in **Photo 4**. In fact, the LED display electronics shown in **Photo 2** are also mounted on a break-away PCB segment.

An ICSP adapter, a MCP9701A temperature sensor, and a five-volt OMRON G6K series relay and supporting MOSFET driver are also standard eM8 equipment available via break-away PCBs. The 10-pin ICSP connection shown in **Photo 3** also includes I^2C and UART interface points in addition to the programming/debugging pins. All of the current Microchip programming/debugging products can be used to program and debug the eM8's PIC16F882. The supported programming/debugging devices include the PICkit2, PICkit3, ICD2, ICD3, and Real ICE. The programming adapter, temperature sensor, and relay are shown still strung together in **Photo 5**.

7.5 and 25 volts AC or DC. A full wave bridge rectifier mounted just below the power jack in **Photo 1** guarantees the correct DC input polarity and rectifies any AC input voltage. Be careful if you use an AC source supply as you must consider that applying 24 VDC is much different than applying 24 VAC at the eM8's power input jack. A nine-volt DC wall wart is safe and our power source of choice.

The eM8 design reserves the least significant nibble of the PIC16F882's PORTA for analog use. AN0 and AN1 have variable gain Microchip MCP6032 op-amp front ends. The op-amps are configured to provide gains that range from unity to 10. The analog circuitry can be seen just to the right of the 7805 voltage regulator in **Photo 3**.

On the digital side, eight digital I/O pins are supported in the original eM8 application. The I/O pins can also be routed to buttons and LEDs which are located on the break-away segment of the eM8 main printed circuit board (PCB)

There is plenty of information about the eM8 hardware that can be ascertained from simple observation and the eM8 Specifications Summary document. For instance, I verified that SOT-23-packaged parts marked as 12W are MOSFETs and SOT-23-packaged parts marked as A6t are diodes by using the Summary's description of the break-away relay PCB. A pair of A6t diodes are connected in parallel and attached in parallel with the G6K relay coil. A 12W MOSFET looks to be switching the ground side of the G6K's relay coil.

The more I stared at the eM8, the more I learned about it. The LM317 adjustable power supply circuitry is standard stuff and so is the 7805 voltage regulator design. I'm really tempted to pull out the ohmmeter and start tracing through the CMOS latch and buffer circuitry. That's not going to happen since once we dive into the source code, I'm sure that many more of the eM8's hardware secrets will be revealed.

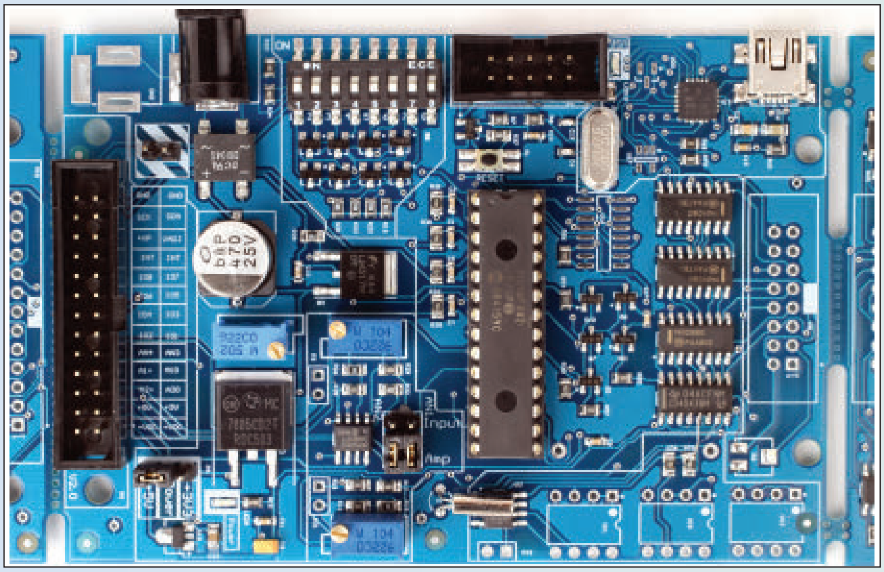


PHOTO 3. Stare at this long enough and the eM8's major electronic subsystems will magically appear.

It's Written in Assembler!!

That it is. The eM8 source code was composed with PIC assembler source. I am not surprised as the eM8 and its code are intended as teaching tools. If you've never tried writing a PIC program using assembler mnemonics, you haven't lived. The cool thing here is that if you have reservations about PIC assembler and/or C, you'll be

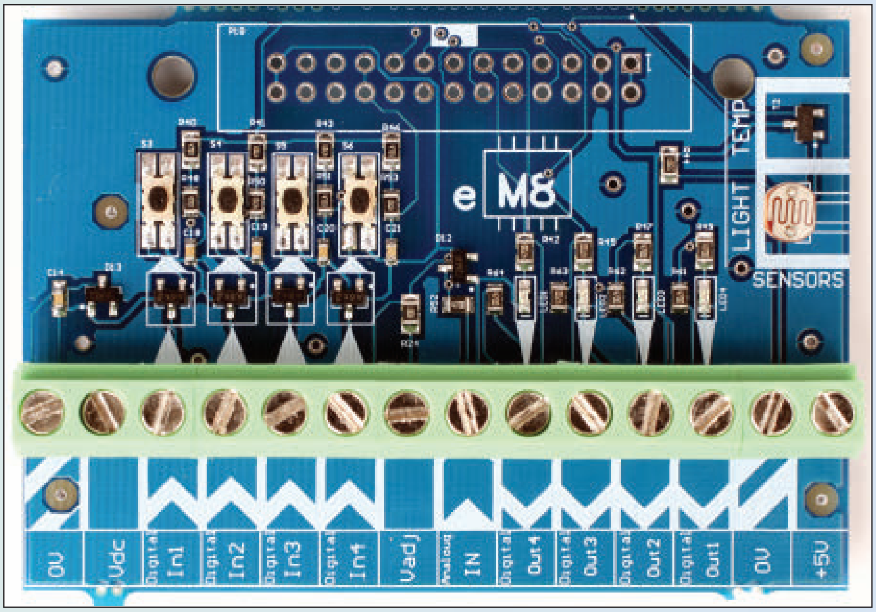


PHOTO 4. The screw terminal legend pretty much tells the story behind this break-away portion of the eM8. The legend for the I/O break-away module is located just to the right of the 26-pin connector in Photo 3.

well versed in both by the time we finish the eM8 assembler-to-C port. Who knows, you might even find a robotic application for the eM8.

Duplicate the Definitions

To be successful, we must pay attention to detail and take whatever clues about the code and hardware we can get. At the very top of the source code, Sam has provided what I call a definition schematic:

```
;             Processor Pin Assignments
;                    ________\/________
;     MCLR -|1 MCLR                  28  |- ISPDAT
;     AN1  -|2 AN0                   27  |- ISPCLK
;     AN2  -|3 AN1          RB5/nT1G 26  |- IO8
;     AN3  -|4 AN2/Vref-     B4/P1D  25  |- IO7
;     AN4  -|5 AN3/Vref+  RB3/AN9    24  |- IO6
;     IO1  -|6 RA4             RB2   23  |- IO5
;     IO2  -|7 RA5             RB1   22  |- SO_SEL--
;     GND  -|8 Vss          RB0/INT  21  |- INT_IO
                                              RB0
;     IO3  -|9 RA7             Vdd   20  |- +5V
;     IO4  -|10 RA6            Vss   19  |- GND
; <>  SO0  -|11 RC0          RC7/RX  18  |- RX
; <>  SO1  -|12 RC1          RC6/TX  17  |- TX
; <>  SO2  -|13 RC2            RC5   16  |- SO3 <>
;     SCL  -|14 RC3/SCL   RC4/SDA    15  |- SDA
;                  ______________
```

Sam's assembler equates are associated with the PIC16F882's actual I/O pin layout. The comments that follow the definition schematic tell us that the PIC16F882 is running on its 8 MHz internal oscillator. We are also told that TIMER 0 is used as the system timer and is running at 1 kHz. Here's the rest of the configuration story as told by the assembler code:

```
__CONFIG _CONFIG1, _LVP_OFF &
       _FCMEN_OFF &
       _IESO_OFF &
       _BOR_OFF &
       _CPD_OFF &
       _CP_OFF &
       _MCLRE_ON &
       _PWRTE_ON &
       _WDT_OFF &
       _INTRC_OSC_NOCLKOUT

__CONFIG _CONFIG2, _WRT_OFF &
       _BOR40V
```

The eM8 original fuse configuration is easily duplicated by employing the services of the CCS C compiler Project Wizard. All we have to do to port the fuse settings is specify identical configuration fuse definitions. Here's the CCS C compiler version of the fuse configuration:

```
#include <16F882.h>
#device ICD=TRUE
#device adc=8

#FUSES NOWDT          //No Watch Dog Timer
#FUSES INTRC_IO       //Internal RC Osc, no
                      //CLKOUT
#FUSES PUT            //Power Up Timer
#FUSES MCLR           //Master Clear pin enabled
#FUSES NOPROTECT      //Code not protected from
                      //reading
#FUSES NOCPD          //No EE protection
#FUSES NOBROWNOUT     //No brownout reset
#FUSES NOIESO         //Internal External Switch
                      //Over mode disabled
#FUSES NOFCMEN        //Fail-safe clock monitor
                      //disabled
#FUSES NOLVP          //No low voltage prgming,
                      //B3(PIC16) or B5(PIC18)
                      //used for I/O
#FUSES NODEBUG        //No Debug mode for ICD
#FUSES BORV40         //Brownout reset at 4.0V
#FUSES NOWRT          //Program memory not write
                      //protected

#use delay(clock=8000000)
#use rs232(baud=9600,parity=N,xmit=PIN_C6,
rcv=PIN_C7,bits=8)
```

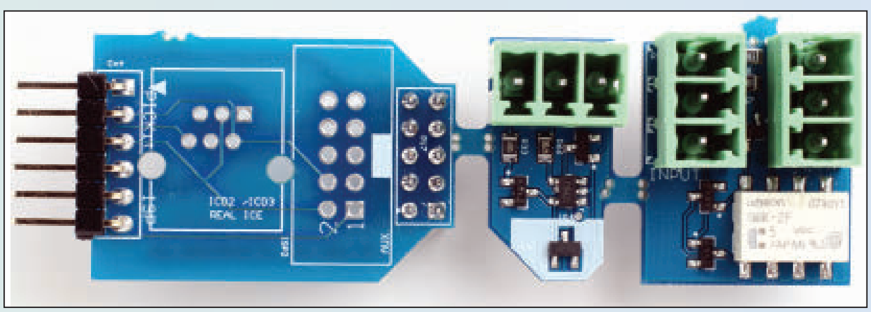


PHOTO 5. From left to right: the ICSP programming/debugging adapter, the Microchip MCP9701A temperature sensor, and the OMRON G6K relay board.

We won't worry about the RS-232 baud rate as we don't know what speed has been defined at this point. The same goes for the ICD and analog-to-digital converter declarations.

The PIC16F882's general-purpose registers begin at address 0x20 in Bank 0 and so do the eM8's variable registers which are the same thing as SRAM. Here's how each eight-bit memory location is allocated in assembler:

```
; Bank 0 address
    CBLOCK 0x20 ;Define GPR variable register
    locations
       PAB           ;Port A Buffer
       PBB           ;Port B Buffer
       PCB           ;Port C Buffer
      CNTL           ; timer low
      CNTH           ; timer high
      CNTDL          ; counter devider low
      CNTDH          ; counter devider high
      gb4
;     gb5            ; general buffers
;     gb6
;     gb7
   ENDC
```

In the case of C, the CCS compiler does all of the hard work. All we have to do is ask for some SRAM space like this:

```
int8  PAB;            //Port A Buffer
int8  PBB;            //Port B Buffer
int8  PCB;            //Port C Buffer
int8  CNTL;           //timer low
int8  CNTH;           //timer high
int8  CNTDL;          //counter divider low
int8  CNTDH;          //counter divider high
int8  gb4;            //general buffers
```

The 96 bytes of general-purpose registers that begin at address 0x20 in Bank 0 end at address 0x7F in Bank 0. There are 32 bytes of SRAM area in Bank 1 between address extents 0xA0 and 0xBF. The PIC16F882 datasheet states that it contains 128 bytes of SRAM; now you know where every byte resides.

A couple of assembler constants are declared next:

```
;********   Constants ********************
CNTDLR         equ     0x64
CNTDHR         equ     0x01
```

If you've had the chance to read my recent CCS C compiler handbook *Master And Command C for the PIC MCU*, the following C statements will come as no surprise to you. Here's the C interpretation of the assembler constants:

```
#define  CNTDLR      0x64
#define  CNTDHR      0x01
```

As you have seen, the programmer must keep up with SRAM usage when writing the code using assembler. Let's work through this set of assembler SRAM allocations:

```
;********   Access REG   **********************
; Access registers are used by several calls
; and services
DBL         equ  0x71   ; Data pointer Low
DBH         equ  0x72   ; Data Pointer High
APL         equ  0x73   ; Address Pointer
                        ; Low
APH         equ  0x74   ; Address pointer
                        ; High
UF          equ  0x75   ; User Flags
CF          equ  0x76   ; Control Flages
DS_READ     equ  0x77   ; input from dip switches
gb0         equ  0x78   ; temp buffers
gb1         equ  0x79
gb2         equ  0x7A
gb3         equ  0x7B
DPOINT      equ  0x7C
FSR_Store   equ  0x7D
w_temp      equ  0x7E   ; variable used for
                        ; interrupt context
                        ; saving
status_temp equ  0x7F   ; variable used for
                        ; interrupt context
                        ; saving
```

I've intentionally eliminated the user flags and control flags since we'll create C structures to hold the flag bits. I also didn't include the w_temp and status_temp allocations as C takes care of saving and restoring the interrupt context information. I've chosen (for now) to keep the FSR_Store although I have a gut feeling we'll be nixing that entry, as well. Here's the initial pass at the C port:

```
//Access REG
int8  DBL;               //Data pointer Low
int8  DBH;               //Data pointer High
int8  APL;               //Address pointer Low
int8  APH;               //Address pointer High
int8  DS_READ;           //input from DIP switches
int8  gb0,gb1,gb2,gb3;   //temp buffers
int8  DPOINT;
int8  FSR_Store;
```

The bits within the UF and CF SRAM locations are the next elements to be defined in the assembler source:

```
;*****   User Flags BIT Def   UF
NU      equ    0x0
DBu     equ    0x1    ; debounce control buffer
TK      equ    0x2    ; tick set from timer 0
SW_CH   equ    0x3    ;change in reed switch
BCD0    equ    0x4    ;BCD Display Counter
BCD1    equ    0x5    ;BCD Display Counter H
TPWM    equ    0x6    ;Display and read dip
                      ;switch off
ADCR    equ    0x7    ;AD New Read
;****** Control Flages  CF

FQ      equ    0x3    ;used for FQ mode select
TRG     equ    0x4    ;trigger bit used to store
                      ;last trigger
CNT_ON  equ    0x5    ;counter on bit counter
```

Sources

Queensland University of Technology
eM8
(Please contact Sam Wallace at: eM8@qut.edu.au for contact information and product details.)

CCS
CCS C compiler
www.ccsinfo.com

```
                ;timer on off control
AD_DIP equ    0x6    ;selects if DIP is read
                ;for input select (Bits 1
                ;and 2)
AD_JUS equ    0x7    ;set right if set
```

We could have easily kept the assembler syntax and tested bits within the UF and CF SRAM locations. In that case, we would use the CCS C compiler's built-in functions *bit_clear*, *bit_set*, and *bit_test*. For instance, here's the C code to set UF flag bit ADCR:

```
bit_set(UF,ADCR);
```

To test the ADCR bit, we use the built-in function bit_test:

```
value = bit_test(UF,ADCR);
```

Clearing and setting the ADCR bit using assember looks like this:

```
bcf    UF,ADCR
bsf    UF,ADCR
```

On the other hand, we can make the C source code a bit easier to follow by rounding up all of the flag bits into their respective UF and CF corrals using special C structures:

```
//User Flags
typedef struct{
    int8  NU:1;
    int8  DBu:1;       //debounce control buffer
    int8  TK:1;        //tick set from timer 0
    int8  SW_CH:1;     //change in reed switch
    int8  BCD0:1;      //BCD Display Counter
    int8  BCD1:1;      //BCD Display Counter H
    int8  TPWM:1;      //Display and read dip
                       //switch off
    int8  ADCR:1;      //AD New Read
}UFFLAGS;

//Control Flags
typedef struct{
    int8  na:3;
    int8  FQ:1;        //used for FQ mode select
    int8  TRG:1;       //trigger bit used to store
                       //last trigger
    int8  CNT_ON:1;    //counter on bit counter
                       //timer on off control
    int8  AD_DIP:1;    //selects if DIP is read
                       //for input select (Bits
                       //1 and 2)
    int8  AD_JUS:1;    //set right if set
}CFFLAGS;
```

Once the UF and CF structures are defined, we must instantiate an instance of each of the structures before we can use them. Here's the code:

```
UFFLAGSUF;
CFFLAGSCF;
```

Now, using the CCS compiler, we can set, clear, and test user and control flag bits like this:

```
UF.ADCR = 1;           //set ADCR bit
CF.CNT_ON = 0;         //clear CNT_ON bit
```

```
if(CF.AD_DIP == 1)
{
        //do something here
}
```

Here are the last of the assembler equate statements in the *eMate.asm* file we've been porting:

```
;*****   Display Bit Def
SO0     equ            0x0    ; BCD bit 0  RC0
SO1     equ            0x1    ; BCD bit 1  RC1
SO2     equ            0x2    ; BCD bit 2  RC2
SO3     equ            0x5    ; BCD bit 3  RC5
SO_SEL  equ            0x1    ; RB1 Control for
                              ; DIP or display
                              ; 1 = Display
;******* COM PORT *****

SRX     equ            0x6    ; Display Select
                              ; 0 RC6
STX     equ            0x7    ; Display Select
                              ; 1 RC7

;******** Button Locations
But1    equ            0x2    ; location of
                              ; buttons on port b
But2    equ            0x3
But3    equ            0x4
But4    equ            0x0
```

I think you have an idea of what the equivalent C source will look like:

```
// Display Bit Def
#define SO0     0x00    //BCD bit 0 RC0
#define SO1     0x01    //BCD bit 1 RC1
#define SO2     0x02    //BCD bit 2 RC2
#define SO3     0x05    //BCD bit 3 RC5
#define SO_SEL  0x01    //RB1 Control for DIP or
                        //Ddisplay 1 = Display
//COM Port
#define SRX     0x06    //Display Select 0 RC6
#define STX     0x07    //display Select 1 RC7
//Button Locations
#define But1    0x02    //location of buttons on
                        //port b
#define But2    0x03
#define But3    0x04
#define But4    0x00
```

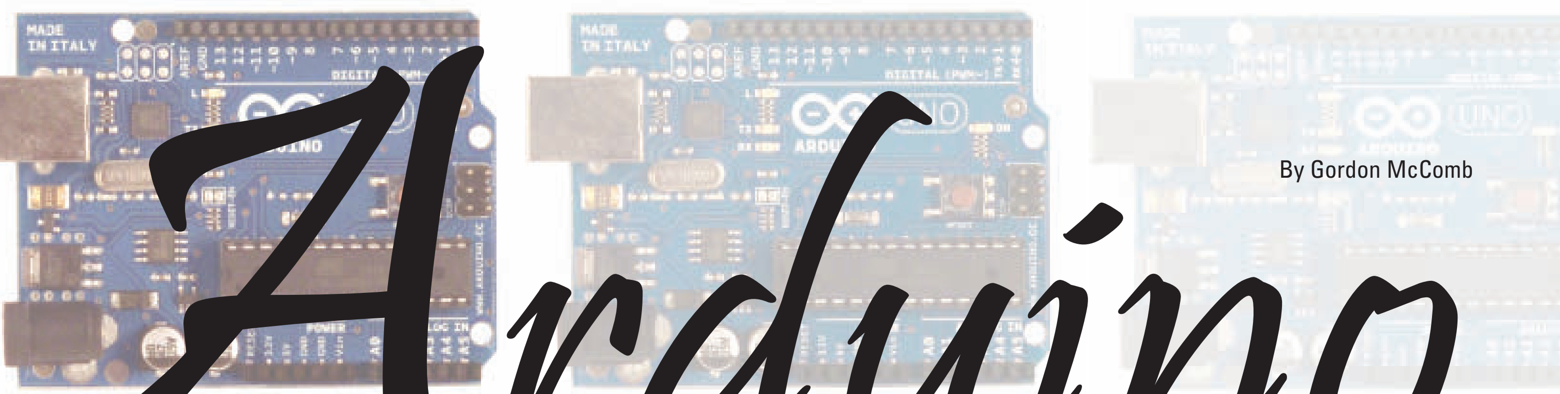
What's Next?

We've ported what I believe to be one of the most important parts of any program — the initial definitions. The next set of assembler code we will tackle is the application setup portion of the *eMate.asm* file. The eM8 application is made up of a number of peripheral-specific include files which are called from *eMate.asm*. So, as we move through the *eMate.asm* code, we'll branch off and port dependent peripheral-specific include files as we come to them. If there is anything exciting we can do with our ported code and the eM8 hardware along the way, you can count on us getting out of the truck and walking down those trails. SV

Fred Eady can be reached via email at fred@edtp.com.

Making Robots With The Arduino

Part 1



By Gordon McComb

Twenty years ago, I began work on my ultimate home robot. Its brain was an Intel 80286-based PC motherboard, running at a whopping 8 MHz. The robot used a floppy disc drive to load the operating system and programs, and custom prototype boards for external interfacing.

The beast needed a hefty battery for power, and with the battery alone weighing some 15 pounds, I needed a sturdy frame to keep everything together. Constructed of aluminum, the robot measured 18 inches square by almost three feet high, and required heavy duty and expensive gear motors — all this just to meander down the hallway and scare the &@%! out of my cat.

Five years and over $1,500 later, I put "Maximillian" to rest, pulling its parts to use in other projects. Robot electronics were shrinking, and that meant robots themselves were getting smaller. Innovations like the BASIC Stamp made it much easier to experiment with low cost, self-contained microcontrollers — perhaps the ideal robotic brain. Microcontrollers are now so commonplace that you have your pick of hundreds of makes and models; from the super simple, to the confoundedly complex. Somewhere in the middle is the Arduino — a small and affordable microcontroller development board that's fast becoming something of a superstar.

Why the Popularity?

Sure, the Arduino is a capable little critter able to handle the most common things microcontrollers can do. And let's not forget that some of its fame has to do with price: the standard Arduino costs about $30, assembled and tested. Even less if you want to build it from a kit.

Then there's its free programming software. Using a standard USB cable, it lets you easily connect the Arduino to your computer — Windows, Mac, or Linux — and begin working in minutes. The programming editor is simple to use and comes with several dozen examples to get you started.

What's really made the Arduino a darling of geeks the world over is this: Both its hardware design and software are open source. That means others are able to take the best ideas and improve on them, all without paying licensing fees. This has created something of a cottage industry of fans and third-party support.

Though the most popular version of the Arduino is made by a company in Italy (where the board was originally developed), many others offer compatible designs in one form or another. Add to this a growing body of add-ons that maximize the Arduino, and free resources for programming examples, code libraries, and step-by-step tutorials.

Introducing Arduino Robotics

So, it makes sense to look at ways to leverage the Arduino to build robots. That's exactly what we'll be doing in this article and several more to follow in the months ahead. I'll show you how to build, program, and use an economical and expandable autonomous desktop robot — the ArdBot — that's powered by an Arduino. Cost of the project is under $85 — even less if you already have some basic components like a solderless breadboard and hookup wire.

The robot base is simple to build and can be constructed out of a variety of materials; no special tools are required. I'll demonstrate a version made of expanded PVC plastic, but you can use heavy cardboard, foam board, picture frame mat board, or most any other material you like. (For your convenience, you can get the robot chassis precut with all the hardware; see the **Sources** box for more information.)

I believe in robot designs that let you explore and experiment, and the ArdBot leaves plenty of room for expansion and independent discovery. You can use the robot for line or wall following, maze solving, or general meandering around in a room. (Cat scaring optional.) You can also take the concepts presented here and design your own version of the ArdBot — bigger or smaller, wheels or tracks — your choice.

In this installment, you'll learn all about the Arduino: what it's made of, how to connect it to your computer, and how to start developing robot projects for it. You'll also be introduced to the ArdBot chassis, including where to get its main parts. In coming installments to this series, you'll explore programming the robot to do interesting things, and extending its features with sensors and other add-ins.

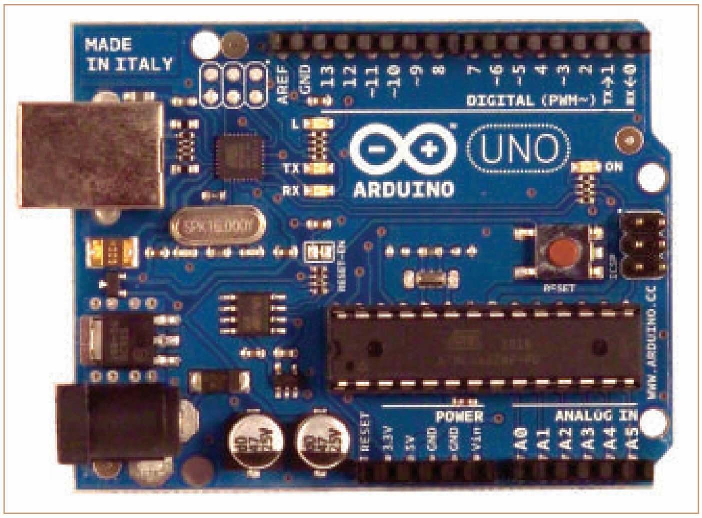


FIGURE 1. The Arduino Duo is a compact microcontroller board based on the Atmel ATmega328 chip. It's available from a number of sources at about $30 average retail.

Arduino Under the Hood

First introduced in 2005, the Arduino has gone through numerous iterations, revisions, and improvements. As I'm writing this, the Arduino team just released their newest version: the Arduino Uno (see **Figure 1**). Like its predecessors, the Uno is an all-in-one development board. It contains an Atmel AVR microcontroller — specifically the ATmega328 — a USB-to-serial interface, five volt voltage regulator, and various support electronics.

Previous iterations of the Arduino have included the Duemilanove (which means 2009 in Italian) and the Diecimila which means 10,000 (a reference to the number of Arduino boards that had been manufactured by that time; many more have been made since).

The Uno, Duemilanove, and Diecimila are what might be called main or core board designs. These all share a common form factor which is a PCB that measures 2-1/8" by 2-3/4". All contain a power jack for a 2.1 mm (center positive) barrel connector, as well as a USB Type B jack for hooking up to a host computer.

A series of 28 female pin headers allow connection of external devices to the Arduino. The headers are separated into three groups as shown in **Figure 2**. The groups are: power, analog input, and digital input/output. Of the 28 pins, 20 are devoted to input and output. There are six analog input pins which can also serve as general-purpose digital I/O. The 14 digital input/output pins include six that can be used to generate PWM (pulse width modulated) signals; these are useful for such things as controlling the

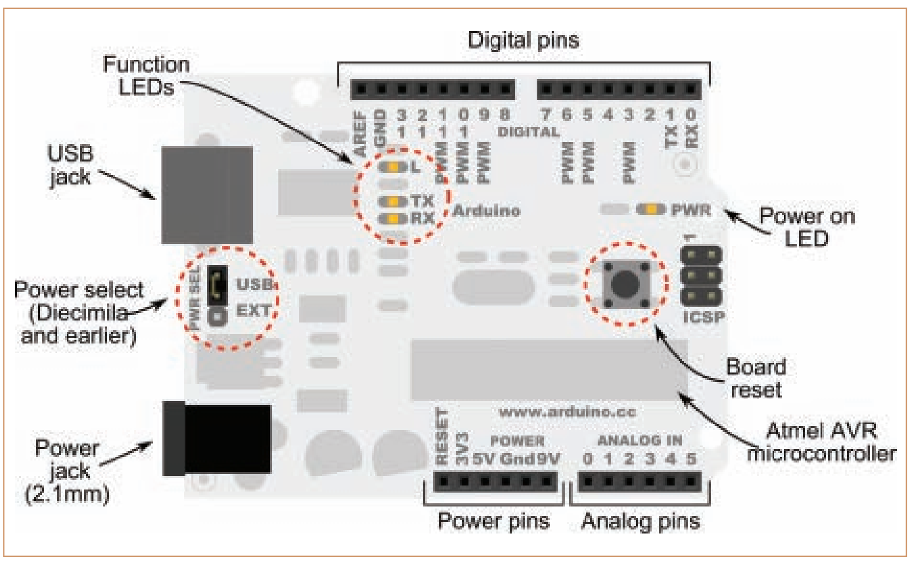


FIGURE 2. Points of interest on the Arduino board include the USB and power jacks, function and power LEDs, and rows of connection headers.

Table 1. Arduino at a Glance.		
Arduino Version	Microcontroller	Supports Standard Expansion Shields
Uno and Duemilanove (2009b)	ATmega328	Yes
Duemilanove (pre-2009b) and Diecimila	ATmega168	Yes
Mega 2560	ATmega2560	No
Nano, Mini, LilyPad, others	ATmega168 or ATmega328	No

Table 2. Microcontroller Specifications.			
	ATmega168	ATmega328	ATmega2560
Flash memory	16 KB; 2 KB used by bootloader	32 KB; 0.5 KB used by bootloader	256 KB; 8 KB used by bootloader
SRAM	1 KB	2 KB	8 KB
EEPROM	512 bytes	1 KB	4 KB
Clock speed	16 MHz	16 MHz	16 MHz

speed of motors. Through its I/O pins, the Arduino supports the basic inter-communications standards: TTL serial, SPI, 1-Wire, and I²C. Two of its pins (digital I/O lines 2 and 3) support hardware interrupts that via software trigger on a LOW value, a rising or falling edge, or a change in value.

Like any microcontroller, the Arduino is basically a small single-board computer designed to interface to external hardware like switches, motors, lights, relays, sensors, and LEDs. At the heart of the Arduino is an Atmel AVR microcontroller. The exact version of AVR controller depends on the version of the Arduino. For example, the older Diecimila and the first Duemilanove versions used an AVR ATmega168; the second generation Duemilanove (referred to as 2009b) as well as the Uno, use the AVR ATmega328. The '328 is physically identical to the '168 but it contains more memory space. See **Tables 1** and **2** for details on variations between the controller chips used.

The bulk of the components on the Arduino are surface-mount, but on most Arduino boards the AVR microcontroller is provided in a dual inline pin (DIP) package. This permits easy replacement should that ever be needed. A new AVR chip costs maybe $5 or $6; that's a lot cheaper than replacing the entire Arduino board.

Keep in mind that the AVR provided in commercially manufactured Arduino boards comes with a bootloader program pre-installed in its Flash memory. This bootloader allows you to program the Arduino by using a USB connection to your PC. When replacing the AVR microcontroller of an Arduino, you need to either purchase a chip with the bootloader software pre-installed, or if you have the proper hardware setup — like an Atmel STK500 programmer — you can do it yourself. Instructions for downloading bootloader software into an AVR chip are provided on the main Arduino information page.

Table 3. Arduino Pin Resources.	
Arduino Uno, Duemilanove, and Diecimila	
Digital I/O Pins	14 (of which six provide PWM output)
Analog Input Pins	6
Nano	
Digital I/O Pins	14 (of which six provide PWM output)
Analog Input Pins	8
Mega 2560	
Digital I/O Pins	54 (of which 14 provide PWM output)
Analog Input Pins	16

Many Variations on a Theme

The core board designs of the Uno, Duemilanove, and Diecimila are perhaps the most common and popular of the Arduinos, but there are numerous other variations. Here are just some of the standardized Arduino boards you'll encounter. The Arduino BT and Fio are intended for wireless applications. The BT contains a Bluetooth module; the Fio has a built-in Zigbee radio. (You can also readily add Bluetooth and Zigbee to an Uno or other core board using "shields" detailed below.)

The Nano is a compact stick-shaped board made for breadboard use. It has all the main features of the Uno and others (including built-in USB jack), but measures only 0.73" x 1.7". It uses only surface-mount parts.

The Mini is even smaller, and is ideal for very small bots with limited space. The Mini lacks its own USB jack, and requires the use of a USB adapter or serial TTL connection to the host PC for programming. The Mini has four analog input pins instead of the six or eight of the other versions.

The Mega2560 is based on a larger AVR chip, and it offers over three times the number of analog and digital I/O lines (see **Table 3**). Memory and program space are bigger, too. The Arduino Mega2560 contains 256 KB of Flash (by comparison, the Uno has 32 KB), as well as more RAM and EEPROM space. Use this for the bigger jobs.

Several Arduino resellers (such as Solarbotics and Adafruit) offer their own custom offshoots of the Arduino — these typically go by different names such as Boarduino

or Freeduino to differentiate them from the original Arduino designs. The Adafruit Boarduino (available in kit form for under $18) is like the Arduino Nano. It uses thru-hole components for ease of soldering.

Some variations of the Arduino depart from the standard form-factor of the Uno, and are not designed for use with expansion shields (discussed below). A good example is the LilyPad — a special Arduino layout engineered for making (among other things) wearable microcontroller projects. Think Borg implants, only more friendly looking. The flower-shaped LilyPad has a flat profile and can be sewn into fabric. It has connection points on the ends of its 22 petals.

With so many variations of the Arduino floating around, it's easy to get confused. For the ArdBot, we'll be using an Arduino Duo, but you can readily substitute just about any of the other versions. If you already have an earlier Duemilanove or even Diecimila, you can use it with the ArdBot. The only catch is that you'll need to make sure you have an up-to-date Arduino programming environment installed on your computer. I've tested everything with version 0019 of the Arduino programming IDE (discussed later), so with that version or anything later you should be good to go.

Ready Expansion Via Shields

The Arduino is an example of the KISS principle. Its simple design helps keep costs down, and makes the Arduino a universal development board adaptable to just about anything. While there are more expensive specialty versions of the Arduino made for robotics applications, the basic board lacks connectors to directly attach to motors, sensors, or other devices.

The Arduino itself has no breadboard area, but it's easy enough to connect any of the inputs or outputs to a small breadboard via wires. For an application like robotics, you'll want to expand the Arduino I/O headers to make it easier to plug in things like motors, switches, and ultrasonic or infrared sensors.

One method is to use an add-on expansion board known as a shield. These stick directly on top of the core board designs (Uno, Duemilanove, and Diecimila). Pins on the underside of the shield insert directly into the Arduino's I/O headers. Two popular expansion shields are the solderless breadboard and the proto shield; both provide prototyping areas for expanding your circuit designs.

Of course, you don't absolutely need a shield to expand the Arduino. You can place a breadboard — solderless or otherwise — beside the Arduino, and use ribbon cables or hookup wire to connect the two together. This is the approach we'll be using with the ArdBot described in this series of articles.

USB Connection and Power

To allow the easiest possible means of programming, the Arduino Duo and related core boards support on-board USB. You merely need to connect a suitable USB cable between the Arduino and your computer. The cable even provides the power to the board. The necessary USB drivers are provided with the Arduino software. In most cases, installation of the drivers is not fully automatic, but the steps are straightforward and the Arduino support pages provide a walk-through example.

The Arduino accepts a standard USB Type B connector. Your PC probably uses the larger Type A connector, so you need a Type A to Type B USB cable. Keep in mind that some PCs and laptops may use Mini-A or Mini-B connectors, so check first before purchasing a cable for use with the Arduino.

Operating voltage of the Arduino circuitry is five volts which is supplied either by the USB cable when it's plugged into a USB port on your computer, or by a built-in linear regulator when the board is powered externally. The regulator is intended to be powered by 7-12 VDC; a nine

Main Components

This is a *selected* list of North American sources for the main components for the ArdBot.

Arduino Duo or Duemilanove

Source	**Item or SKU**
Adafruit	50
HVW Tech	28920 (Freeduino SB)
RobotShop	RB-Ard-03
SparkFun	DEV-09950

Solderless breadboard, 170 tie-points

Source	**Item or SKU**
Adafruit	65
HVW Tech	21380
Parallax	700-00012
RobotShop	RB-Spa-139

Continuous rotation servo (Futaba spline)

Source	**Item or SKU**
Parallax	900-00008
Pololu	1248
RobotShop	RB-Gws-23
Solarbotics	36000
SparkFun	ROB-09347

2-1/2" or 2-5/8" rubber wheels (Futaba spline)

Source	**Item or SKU**
Adafruit	167
HVW Tech,	
Solarbotics	SW
Parallax	28109
Pololu	226
RobotShop	RB-Sbo-86

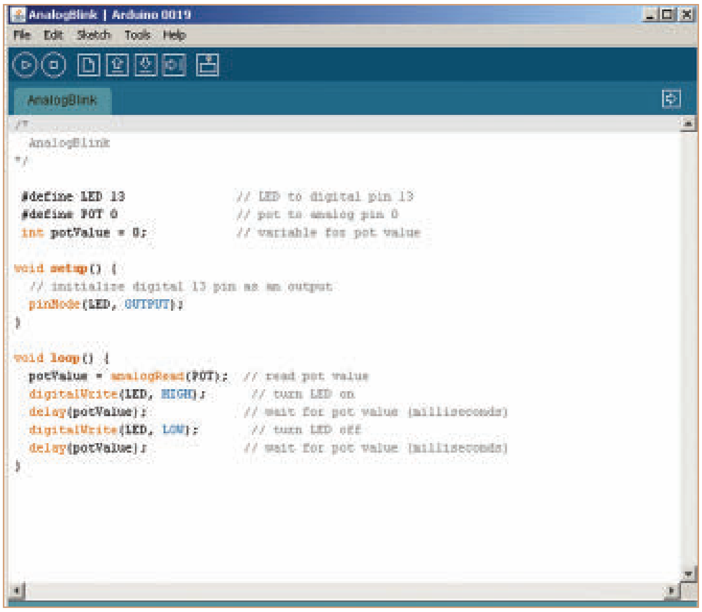


FIGURE 3. The Arduino integrated development environment (IDE) provides a centralized place to write, compile, and download programs to the Arduino board.

volt battery is ideal. Anything higher than 12 volts is not recommended as it could cause the regulator to overheat.

For robotics, I think it's best to power the Arduino from its own battery. The ArdBot uses a split supply where the Arduino is powered by a nine volt transistor battery; a separate four-cell AA battery holder is used for servo motors and other components that don't require voltage regulation.

Indicator LEDs are provided on the Arduino for testing and verification. A small green LED shows power; two other LEDs show serial transmit and receive activity and should flash when the board is being programmed from your computer. A fourth LED is connected in parallel with digital I/O line 13 and serves as a simple way to test the Arduino and make sure it is working properly. We'll use this feature in a simple example later on in this article.

Programming the Arduino

Microcontrollers depend on a host computer for developing and compiling programs. The software used on the host computer is known as an integrated development environment, or IDE. For the Arduino, the development environment is based on the open source Processing platform (**www.processing.org**) which is described by its creators as a "programming language and environment for people who want to program images, animation, and interactions."

The Arduino programming language leverages an open source project known as Wiring (**wiring.org.co**). The Arduino language is based on good old-fashioned C. If you are unfamiliar with this language, don't worry; it's not hard to learn, and the Arduino IDE provides some feedback when you make mistakes in your programs.

Sources

Adafruit Industries
www.adafruit.com
Arduino resellers and custom shield projects. Convenient premade nine volt battery clip and 2.1 mm barrel connector (see product #80), and nine volt battery holder with switch (product #67).

Arduino
www.arduino.cc
The main Arduino site provides downloads, tutorials, references, design schematics, and other information useful for learning about and using the Arduino family of boards.

Atmel
www.atmel.com/products/AVR
Manufacturers of the AVR microcontrollers used in the Arduino. See their site for datasheets (in PDF format).

Budget Robotics
www.budgetrobotics.com
Custom machined decks, servo mounting hardware, and assembly hardware for the ArdBot.

Freeduino
www.freeduino.org
Home of the Freeduino collaborative project.

HVW Technologies
www.hvwtech.com
Reseller of Arduino products and manufacturer (with Solarbotics) of the *Freeduino SB*.

Parallax
www.parallax.com
Not resellers of Arduino, but they offer continuous rotation servos, wheels, and sensors.

Pololu
www.pololu.com
Wheels, continuous rotation servo motors.

RobotShop
www.robotshop.ca (Canada); *www.robotshop.us* (US)
Full service retailer carrying most all of the official Arduino lineup, plus servo motors, solderless breadboards, and sensors.

Solarbotics
www.solarbotics.com
Continuous rotation servos, five-cell AA battery packs with attached 2.1 mm barrel connector, Arduino, and Arduino-clone boards.

SparkFun Electronics
www.sparkfun.com
Reseller of the Arduino and manufacturer of custom Arduino-like hardware.

If you've dabbled in Basic, you just need to remember that in C, keywords and variables are case sensitive. Instead of using If/End If, in C, code blocks are grouped together using the { and } (brace) characters. Statements are terminated with a ; (semi-colon) character, rather than just a simple line break. Any other differences, you'll pick up quickly.

To get started with programming your Arduino, first go to: **http://arduino.cc** and then click on the Download tab. Find the platform link (PC, Mac, Linux) for your computer and download the installation file. Step-by-step instructions are provided in the Getting Started section of the Arduino website. Be sure to read through the entire instructions.

Be aware that the main Getting Started section assumes you're using an Arduino Uno, Duemilanove, Nano, or Diecimila board. If you're using another version of the Arduino, be sure to check out its corresponding page on the site.

Once installation is complete, you're ready to try out your Arduino. Start by connecting the board to your PC via a USB cable. If this is the first time you've used an Arduino on your PC, you must install the USB communications drivers, as detailed in the Getting Started guide.

Using the Arduino programming environment is simple. First-time use of the environment requires you to specify the Arduino board you are using, and as necessary, the serial port that is connected to the board (the Arduino's USB connection looks like a serial port to your computer). You may then open an existing example program which is called a sketch in Arduino parlance, and download the program to your board. Or, you may write your own sketch in the IDE editor. **Figure 3** shows the Arduino IDE with a short sketch in the main window.

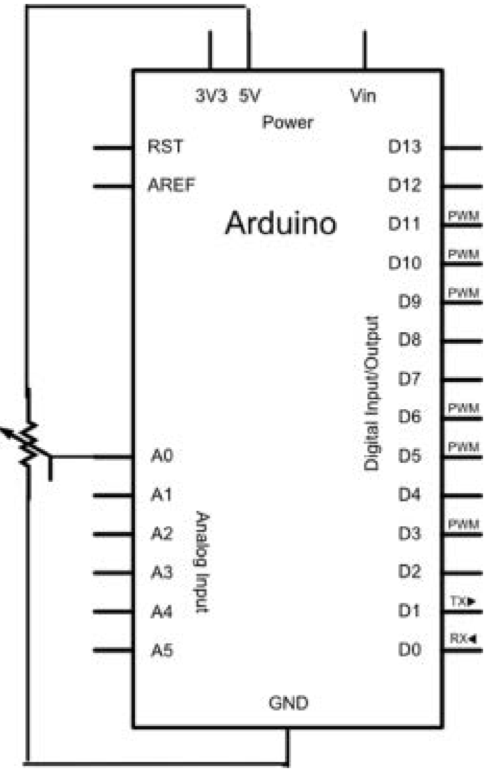
After writing or opening an existing sketch, you need to compile it which prepares the code for downloading to the Arduino. The Arduino IDE calls compiling a program verifying. At the bottom of the text editor is a status window which shows you the progress of compiling (verifying). If the sketch is successfully compiled, it can then be downloaded to the Arduino where it will automatically run once the download is complete.

Programming for Robots

As you go through the list of programming statements available in the Arduino IDE (choose Help->Reference), you might think there isn't much power for doing things like running servos, operating stepper motors, reading potentiometers, or displaying text on an LCD.

Like most any language based on C, the Arduino supports the notion of "libraries" — code repositories that extend core programming functionality. Libraries let you re-use code without having to physically copy and paste it into all your programs. The standard Arduino software installation comes with several libraries you may use, and you can download others from the Arduino support pages and from third-party websites that publish Arduino library code.

FIGURE 4. Schematic for Listing 1 testing circuit.



A good example of a library you'll use with the ArdBot — and likely many other robot projects — is Servo. This library allows you to connect one or more hobby R/C servos to the Arduino's digital I/O pins. The Servo library comes with the standard Arduino installation package, so adding it to your sketch is as simple as choosing Sketch->Import Library->Servo. This adds the line

```
#include <Servo.h>
```

which tells the Arduino IDE that you wish to include the Servo library in your sketch. With the functionality of the library now available to you, you can use its various functions to control one or more servos. For example, you can use the write function to rotate a servo to a specific position, from 0 to 180 degrees. The following code

```
myServo.write(90);
```

moves a servo to its midpoint, or 90 degree position.

Structurally, Arduino sketches are very straightforward and are pretty easy to read and understand. The Arduino program contains two main parts: setup() and loop(). These are programming functions that do what their names suggest: setup() sets up the Arduino hardware, such as specifying which I/O lines you plan to use, and whether they are inputs or outputs. The loop() function is repeated endlessly when the Arduino is operating.

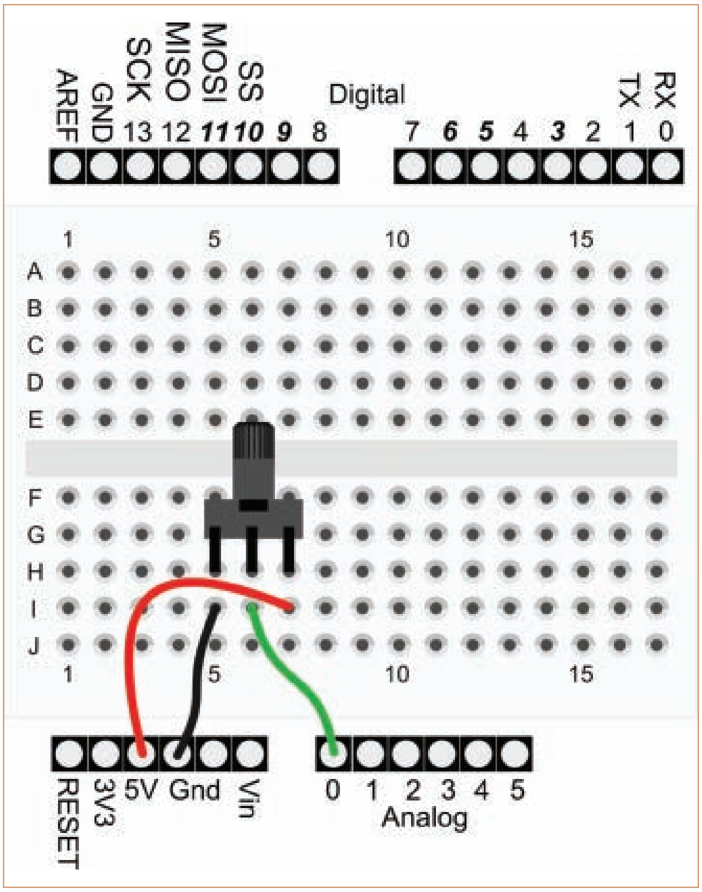


FIGURE 5. Breadboard layout for Listing 1 testing circuit. The potentiometer is 10 kW, linear taper.

Experiment By Doing

Listing 1 demonstrates a few fundamental Arduino concepts useful in any robotics development — that is, reading an analog sensor and providing visual feedback. I've taken one of the examples that comes with the Arduino IDE and modified it slightly to conform to the style we'll be using throughout this series of ArdBot articles. It uses a 10 kΩ potentiometer to alter how fast Arduino's built-in LED flashes.

Listing 1

```
 #define LED 13       // LED to digital pin 13
 #define POT 0        // pot to analog pin 0
 int potValue = 0;    // variable for pot value

void setup() {
  // initialize digital 13 pin as an output
  pinMode(LED, OUTPUT);
}

void loop() {
  potValue = analogRead(POT);       // read pot
                                    // value
  digitalWrite(LED, HIGH);   // turn LED on
  delay(potValue);           // wait for pot
value
                             // (milliseconds)
  digitalWrite(LED, LOW);    // turn LED off
  delay(potValue);           // wait for pot
value (milliseconds)
}
```

Check out **Figure 4** for a schematic of the circuit used for the program listing; **Figure 5** shows a pictorial breadboard view of connecting the potentiometer to the Arduino hardware. The potentiometer is connected to the board as a common voltage divider. That way, the Arduino detects the value of the pot as a variable voltage from zero volts (ground) to five volts.

Note that I'm using a standard mini solderless breadboard with 170 tie-points. The breadboard serves as a prototyping area for connecting to various hardware, and is part of the ArdBot. You can use any other size of solderless breadboard, but none of the circuits for the ArdBot will require anything bigger.

Here's how the program works:

The first two lines set variable constants, so hardware connected to the various I/O pins can be referred to by name and not pin number. This is merely for our convenience. The built-in LED is connected to pin 13, and the potentiometer — named POT in our program — is connected to analog pin 0.

Another variable is defined to hold the current value of the potentiometer which will be a number from 0 to 1023. This number is derived from the Arduino's integrated 10-bit analog-to-digital (ADC) converter, and it represents a voltage level from zero volts to five volts.

The setup() section gets the Arduino hardware ready for the rest of the program. When first powered on, all the I/O lines are automatically defined as inputs. The LED pin needs to be an output, however, so that distinction is defined here. The programming statement that changes the function of an I/O line is pinMode. It expects two values (called arguments): the number of the I/O pin — in this case, it's 13 as defined by the LED variable — and whether the pin is an OUTPUT or an INPUT.

The loop() section is automatically started the moment the program has been downloaded to the Arduino. The looping continues until the board is either unplugged, the reset button on the Arduino is pushed, or a new program is loaded into memory. The loop begins by reading the voltage on analog pin 0 — remember, it's defined in the POT variable at the top of the program. The program then turns the LED on, and waits for a period of time defined by the current position of the potentiometer before turning the LED off again.

The waiting period is in milliseconds (thousandths of a second), from 0 to 1023, the range of values from the Arduino's ADC. Very fast delays of about 100 milliseconds or less will appear as a steady light. You'll be able to see the LED flash with longer delays.

Quick View of the ArdBot

Figure 6 shows the prototype ArdBot, made of 6 mm (about 1/4") expanded PVC. In the next installment, I'll provide detailed construction plans, but here's the robot in a nutshell:

- Two 7" "decks" provide generous room for motors, batteries, Arduino, and mini solderless breadboard, as well as future expansion. The top deck is secured by four machine screws to a set of 1-3/4" long aluminum hex standoffs.
- Two standard size continuous rotation servos drive the robot using the differential steering technique where the speed and direction of each motor determines where the robot travels.
- Half-inch wide tires provide traction indoors and out. The wheels measure 2-1/2" in diameter, and directly connect to the servo shafts.
- To keep costs down the ArdBot doesn't use wheel casters or ball transfers. Instead is uses two height-adjustable skids fore and aft which keep the robot level. The skids have rounded bottoms and act just like small rollers.

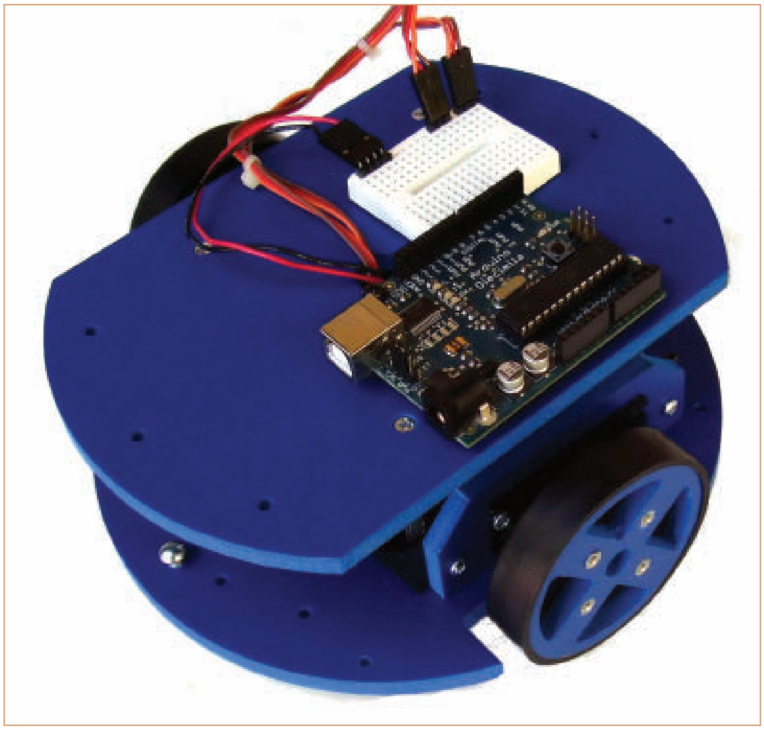


FIGURE 6. Prototype ArdBot, a double-decker desktop robot using the Arduino Duo or similar controller board. It's designed for easy expansion and experimentation. Construction will be covered in Part 2.

The bottom deck is used for mounting the servos and battery packs. The deck is large enough for several four- or five-cell AA battery holders, plus a nine volt cell or custom battery packs. There's room in the corners of the deck for mounting infrared, bump switch, or other sensors.

The top deck provides open access to the Arduino and solderless breadboard, both of which you can place anywhere you want. This way, you can program and reconfigure the board without any disassembly of the robot. There's room for servo turrets, accelerometers, GPS receivers, sensor modules, and more. In the event you need even more room for your experiments, you can add a third deck for an additional 35 square inches of space.

The ArdBot is a universal design with components you can get from a variety of suppliers. See the **Sources** box for a list of online retailers that sell the Arduino and other parts. You can build the ArdBot platform yourself, or if you don't like mechanical construction, as a convenience to *SERVO* readers, you can get the two body decks and all mounting hardware from my Internet company, Budget Robotics.

So much for the basics. See you next time for detailed constructions plans of the ArdBot and more. **SV**

About the Author

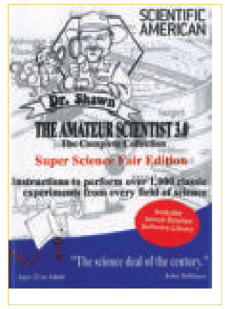
Gordon McComb is the author of *Robot Builder's Bonanza*. He can be reached at arduino@robotoid.com.

The SERVO Webstore

Attention Subscribers ask about your discount on prices marked with an *

CD-ROM SPECIALS

The Amateur Scientist 3.0 The Complete Collection
by Bright Science, LLC

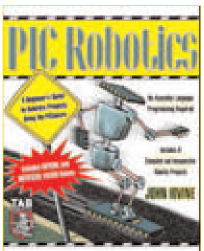


There are 1,000 projects on this CD, not to mention the additional technical info and bonus features. It doesn't matter if you're a complete novice looking to do your first science fair project or a super tech-head gadget freak; there are enough projects on the single CD-ROM to keep you and 50 of your friends busy for a lifetime!

Reg $26.95 Sale Price $23.95

ROBOTICS

PIC Robotics
by John Iovine



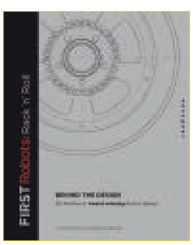
Here's everything the robotics hobbyist needs to harness the power of the PICMicro MCU!

In this heavily-illustrated resource, author John Iovine provides plans and complete parts lists for 11 easy-to-build robots each with a PICMicro "brain." The expertly written coverage of the PIC Basic Computer makes programming a snap – and lots of fun.

$24.95

FIRST Robots: Rack 'N' Roll: Behind the Design
by Vince Wilczynski, Stephanie Slezycki

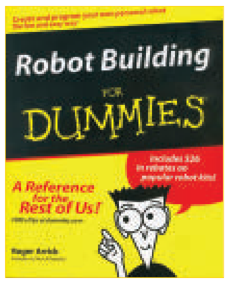
More than 750 photographs!



The second annual book highlighting the creativity and process behind 30 winning robot designs from the 18th annual international FIRST Robotics Competition. The FIRST organization, founded by Dean Kamen (inventor of the Segway), promotes education in the sciences, technology, and engineering.

Reg $39.95

Robot Building for Dummies
by Roger Arrick / Nancy Stevenson

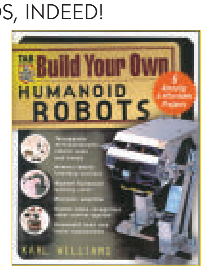


Discover what robots can do and how they work. Find out how to build your own robot and program it to perform tasks. Ready to enter the robot world? This book is your pass-port! It walks you through building your very own little metal assistant from a kit, dressing it up, giving it a brain, programming it to do things, even making it talk. Along the way, you'll gather some tidbits about robot history, enthusiasts' groups, and more.

$24.95

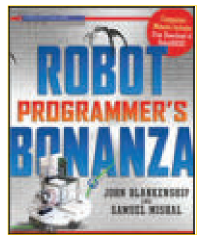
Build Your Own Humanoid Robots
by Karl Williams

GREAT 'DROIDS, INDEED!



This unique guide to sophisticated robotics projects brings humanoid robot construction home to the hobbyist. Written by a well-known figure in the robotics community, *Build Your Own Humanoid Robots* provides step-by-step directions for six exciting proj-ects, each costing less than $300. Together, they form the essential ingredients for making your own humanoid robot. **$24.95***

Robot Programmer's Bonanza
by John Blankenship, Samuel Mishal



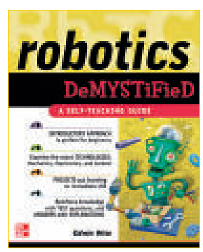
The first hands-on programming guide for today's robot hobbyist!

Get ready to reach into your programming toolbox and control a robot like never before! *Robot Programmer's Bonanza* is the one-stop guide for everyone from robot novices to advanced hobbyists who are ready to go beyond just building robots and start programming them to perform useful tasks.

$29.95

Robotics Demystified
by Edwin Wise

YOU DON'T NEED ARTIFICIAL INTELLIGENCE TO LEARN ROBOTICS!



Now anyone with an interest in robotics can gain a deeper understanding — without formal training, unlimited time, or a genius IQ. In *Robotics Demystified*, expert robot builder and author Edwin Wise provides an effective and totally painless way to learn about the technologies used to build robots! **$19.95**

We accept VISA, MC, AMEX, and DISCOVER
Prices do not include shipping and may be subject to change.

To order call 1-800-783-4624

SERVO Magazine Bundles

Published by T & L Publications, Inc.

$57
per bundle

Save $10 off the normal price!!

Now you can get one year's worth of all your favorite articles from *SERVO Magazine* in a convenient bundle of print copies. Available for years 04, 05, 06, 07, 08, and 09.

Call my Webstore and you'll get someone in AMERICA!

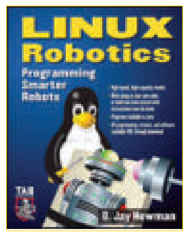
Visit my online store @ www.servomagazine.com

Kickin' Bot
by Grant Imahara

Enter the arena of the metal gladiators!

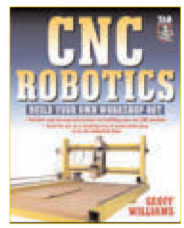
Do you have what it takes to build a battle-ready robot? You do now! Here are the plans, step-by-step directions, and expert advice that will put you in competition — while you have a heck of a lot of fun getting there. Grant Imahara, the creator of the popular BattleBot Deadblow, shares everything he's learned about robot design, tools, and techniques for metal working and the parts you need and where to get them.
$24.95

Linux Robotics
by D. Jay Newman



If you want your robot to have more brains than microcontrollers can deliver — if you want a truly intelligent, high-capability robot — everything you need is right here. *Linux Robotics* gives you step-by-step directions for "Zeppo," a super-smart, single-board-powered robot that can be built by any hobbyist. You also get complete instructions for incorporating Linux single boards into your own unique robotic designs. No programming experience is required. This book includes access to all the downloadable programs you need.
$34.95

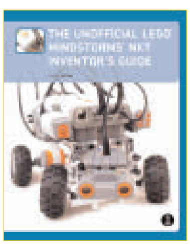
CNC Robotics
by Geoff Williams



Here's the FIRST book to offer step-by-step guidelines that walk the reader through the entire process of building a CNC (Computer Numerical Control) machine from start to finish. Using inexpensive, off-the-shelf parts, readers can build CNC machines with true industrial shop applications such as machining, routing, and cutting — at a fraction of what it would cost to purchase one. Great for anyone who wants to automate a task in their home shop or small business. **$34.95**

SPECIAL OFFERS

The Unofficial LEGO MINDSTORMS NXT Inventor's Guide
by David J. Perdue



This book was written for the first version of the NXT set (#8527), and its projects are only compatible with the first version. In other words, because of piece differences between the NXT 1.0 and 2.0 sets, the projects in this book can only be built with an NXT 1.0 set. However, much of the other information is still helpful, and the building, mechanical, and programming details are still applicable.
Reg $29.95
Sale Price $25.95

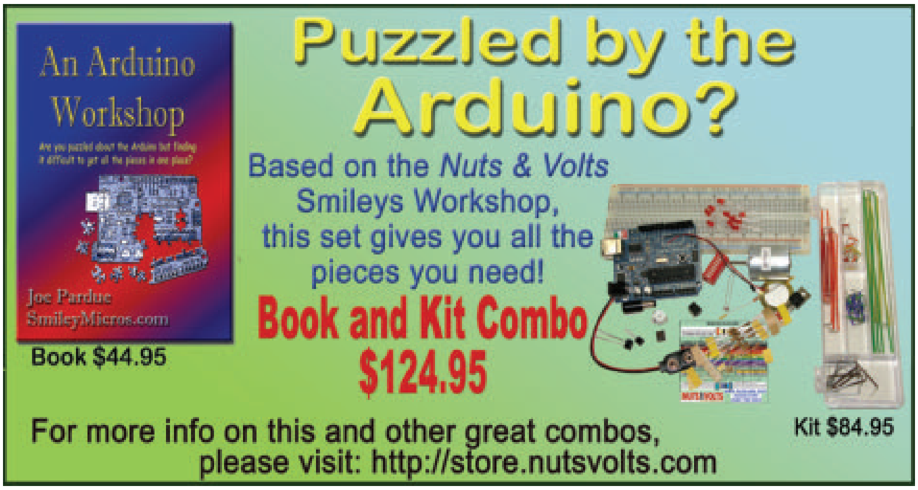
Technology Education Package for Everyone Starting in Electronics

This lab – from the good people at GSS Tech Ed – will show you 40 of the most simple and interesting experiments and lessons you have ever seen on a solderless circuit board. As you do each experiment, you learn how basic components work in a circuit. Along with the purchase of the lab, you will receive a special password to access the fantastic online interactive software to help you fully understand all the electronic principles. For a complete product description and sample software, please visit our webstore.

Regular Price $79.95 **Subscriber's Price $75.95**

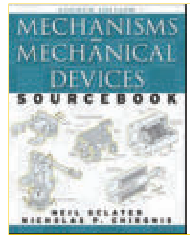
Or order online
www.servomagazine.com

SPECIAL OFFERS



Mechanisms and Mechanical Devices Sourcebook

by Neil Sclater, Nicholas Chironis



Over 2,000 drawings make this sourcebook a gold mine of information for learning and innovating in mechanical design. Overviews of robotics, rapid prototyping, MEMS, and nanotechnology will get you up to speed on these cutting-edge technologies. Easy-to-read tutorial chapters on the basics of mechanisms and motion control will introduce those subjects to you. **Reg $89.95 Sale Price $69.95**

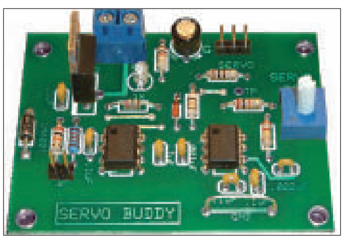
Forbidden LEGO

by Ulrik Pilegaard / Mike Dooley

Forbidden LEGO introduces you to the type of free-style building that LEGO's master builders do for fun in the back room. Using LEGO bricks in combination with common household materials (from rubber bands and glue to plastic spoons and ping-pong balls) along with some very unorthodox building techniques, you'll learn to create working models that LEGO would never endorse. **Reg $24.95 Sale Price $19.95**

PROJECTS

The SERVO Buddy Kit

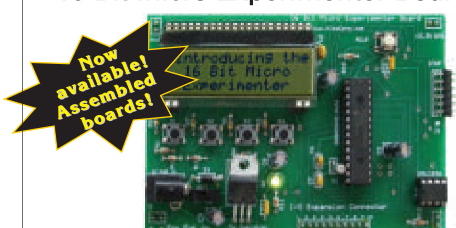


An inexpensive circuit you can build to control a servo without a microcontroller.

For more information, please check out the May 2008 issue or go to the SERVO websore.

Includes an article reprint.
Subscriber's Price **$39.55**
Non-Subscriber's Price **$43.95**

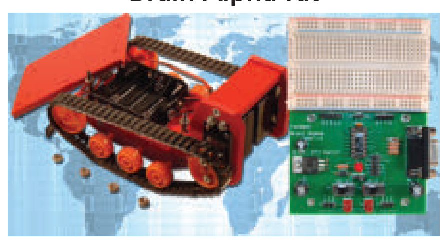
16-Bit Micro Experimenter Board



Ready to move on from eight-bit to 16-bit microcontrollers? Well, you're in luck! In the December 2009 *Nuts & Volts* issue, you're introduced to the 16-Bit Micro Experimenter. The kit comes with a CD-ROM that contains details on assembly, operation, as well as an assortment of ready-made applications. New applications will be added in upcoming months.

Subscriber's Price **$55.95**
Non-Subscriber's Price **$59.95**

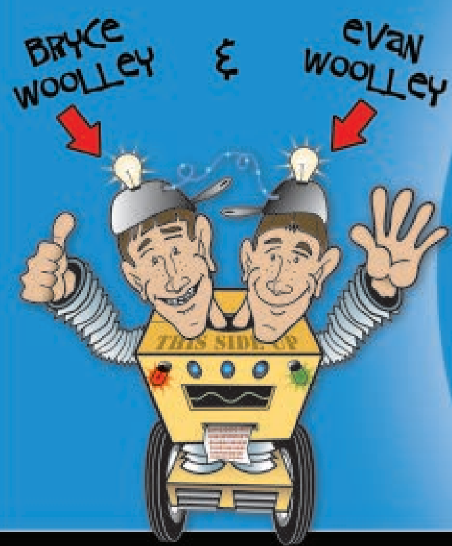
Tankbot Kit & Brain Alpha Kit



Tankbot/Brain Alpha originally by Ron Hackett
Now with New Columnist Calvin Turzillo
A series filled with projects and experiments to challenge you through your learning process while you grow your fully expandable Brain Alpha PCB!
The brain is a PICAXE-14A!
For more info & pictures, visit the *SERVO* Webstore.
Tankbot and the Brain Alpha Kit can be purchased separately.
Combo Price $ 138.95

Twin Tweaks

Bryce Woolley & Evan Woolley



THIS MONTH:

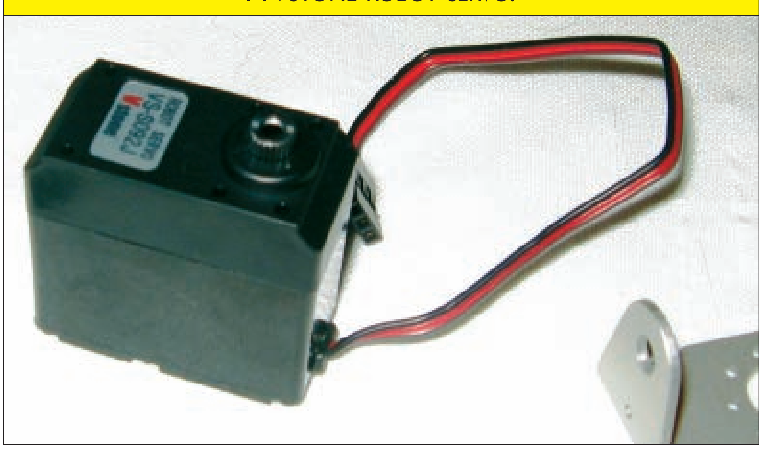
So You Think You Can Dance?

THE KT-X KIT FROM KUMOTEK.

This month, we have the pleasure of presenting the KT-X Bipedal Humanoid Robot from KumoTek. KumoTek is a Texas-based robotics company with products ranging from building inspection robots to hobbyist kits. KumoTek is also quite expert in showmanship; they're the folks behind the RoboSUE exhibit at the Chicago Field Museum. For those that haven't visited the RoboSUE exhibit, it features robotic dinosaurs that interact with museum goers, including RoboSUE the Tyrannosaurus Rex. Even the robots are bigger when they're from Texas.

Having conquered the prehistoric world, KumoTek is jumping into an equally harrowing market – bipedal humanoid servo robots. There are a lot of offerings out there, and the KT-X Superbot strives to distinguish itself as an accessible and expandable platform. With an intuitive method of construction and sophisticated user-friendly programming, we are confident that the KT-X has the potential to dance itself to the top.

A VSTONE ROBOT SERVO.



Dancing With The Parts

The KT-X comes with 17 servos, and thus has 17 degrees of freedom. Other KumoTek models come with different numbers of servos — like 13 for the KT-X Lite and 19 for the KT-X Gladiator. The KT-X comes with a CD choc full of goodies, with everything from the robot software to manuals and a long playlist of motion files.

Roboticists have the option to purchase the KT-X in assembled or unassembled form. We wanted to try our hand at the unassembled kit because having to put everything together gives us a good idea of the bot's motion capabilities and limitations.

The construction manual gives step by step instructions on how to put the robot together. While there may be a slightly rocky start from a bit of tortured translation, the bulk of the instructions are given in clear and detailed diagrams.

Before diving into the assembly, the manual provides a helpful explanation of part notation, revealing that the cryptic names for screws like M2-3 refer to screws 2 mm in diameter with a 3 mm length of threads. Servo humanoid kits are notorious for requiring a plethora of miniature screws, but Kumotek has taken measures to reduce the tedium. In addition to the explanation of the naming scheme, the fact that the screws come in clearly labeled resealable bags is a refreshing example and often overlooked aspect

of user friendliness. The manual also makes the sage suggestion to use a thread locking agent for the screws — something we have been remiss in doing with past servo humanoids that have fallen apart as a result.

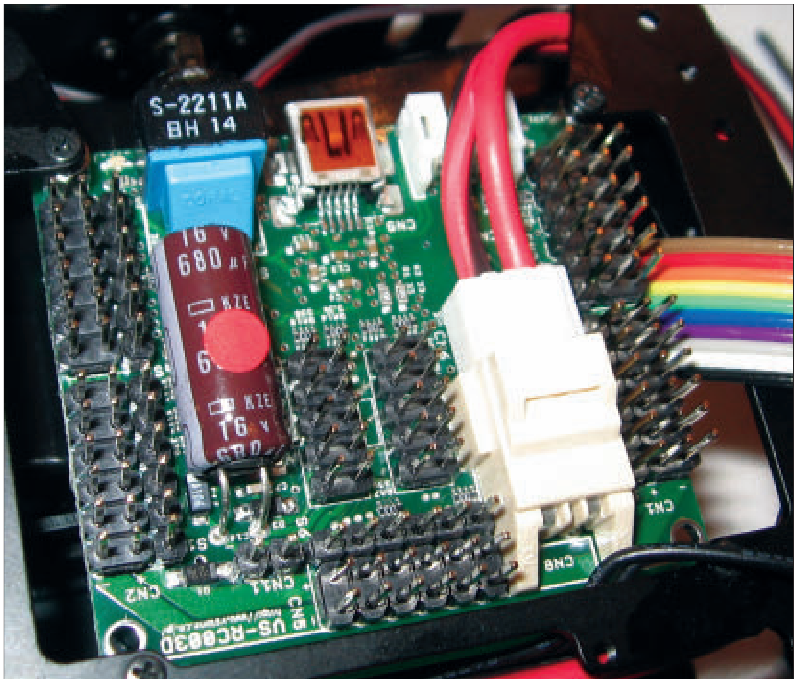
Upon first glance, the basic construction of the KT-X looks quite similar to the construction of other servo humanoids. A closer examination, however, reveals some very helpful design innovations. Firstly, the screws fastening the servos and servo horns to the frames are threading screws, so they cut threads upon assembly for a fit as natural as Billy Elliot in a ballet studio. The aluminum frame pieces are threaded.

The asymmetrical design of the aluminum frame pieces is also more advantageous than the traditional design for the servo brackets we've seen. The traditional design has the frame completely cover the servo horn. While this may allow for more mounting points, it also makes it more difficult to properly line up the mounting holes on the servo horn and the bracket. The KT-X bracket, on the other hand, is easy to position because the exposed part of the horn makes an easy reference.

Before actually putting any of the robot's limbs together, the servos must be calibrated to their zero position. Different kits have different methods for this most foundational of procedures, and we think the KT-X has one of the better solutions. Using the RobovieMaker software and the ports on the CPU for the robot's LED eyes (of all things), each servo is set to the zero position by the software.

After the servos are set to the zero position, the construction of the robot is straightforward. Throughout the process, the manual wisely recommends checking each degree of freedom for a full range of motion. Once the CPU was installed in the body, we really felt like we were getting close. A couple of other electronic bits joined the CPU in the torso, like a speaker and an adapter for a controller.

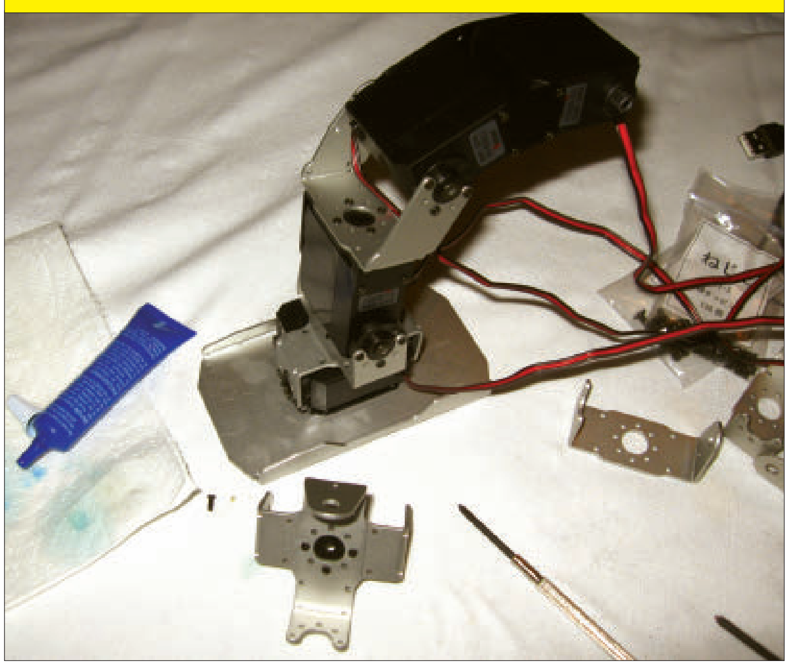
The KT-X uses a wireless game controller. The controller can come with the kit for domestic customers, but the vagaries of the market required us to acquire our own. KumoTek recommends a wireless Logitech Playstation 2 controller. We found an I-Con Playstation 2 controller for a cool $20, but we figured that since the controllers all have to interface with the same hardware that it should be able to work. When we first attempted to install the receiver in the robot, we were distressed when the bulky plastic casing appeared not to fit. We could make the connection between the robot and the receiver, but it appeared that we might have to butcher the front body panel in the process. Before sharpening our aviation shears, however, we tried removing the offending casing. The underlying circuit board fit comfortably into the KT-X, vitiating the need for some frivolous plastic surgery.



THE KTX CPU.

After attaching the limbs to the torso, one of the final steps was to wire up the robot. KumoTek maintains their standard of user friendliness by using servos with shorter wires for joints closer to the torso and servos with longer wires for more distant servos. This is especially helpful when routing the wires. The KT-X uses traditional flexible brackets to bundle the wires, and the excess length near

ASSEMBLING THE KT-X.



Controller receiver complete with bulky casing.

robots is always a bit of a chore with few elegant solutions. Most often, we've seen the wires simply unceremoniously smushed behind the body panel. Given the desire to give the robot a sleek physique this is understandable, but this can make sifting through the wires during troubleshooting or modification a problem. The manual suggests using the ribbon cable from the controller receiver adapter to capture the unruly wires. It results in a snug but secure fit that nicely keeps the wires contained. The ribbon cable from the optional gyroscope module (more on that later) can be used to bundle the wires from the other side of the robot.

After bundling the wires, the body panels go on quite easily. We almost forgot to refrain from thread locking the body panel, but thankfully we remembered that it might be useful to poke around inside it later.

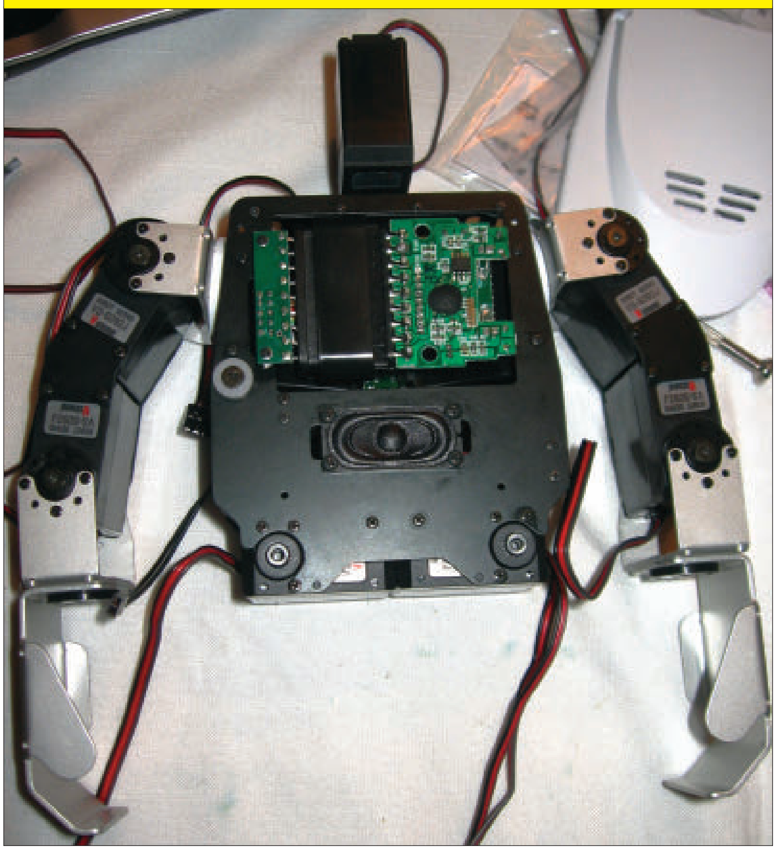
the CPU board is not too terribly unwieldy. As a caveat, some of the brackets are attached to some of the screws previously mounted to the robot frame. Thankfully, we only used blue thread locker and we were able to remove the screws without incident, but it is something to keep in mind.

Dealing with the inevitable wire bundle that accompanies the plethora of servos in bipedal humanoid

Flash Memory Dance

We thought a great way to test the programming capabilities of the robot would be to program in our own custom dance moves. It would be a whimsical way to test the balance of the robot, and it would also give us a recognizable goal to work towards. What pose is more iconic than John Travolta's Saturday Night Fever point, anyway?

Installing the streamlined receiver board.



Firstly, we would like to emphasize how thrilled we are that the KT-X comes with a USB connection to the computer. Not only that, the connection cable is simply a traditional USB Type A and Type B Mini that could be easily replaced if needed. Connecting the robot to the PC required no adapters or trial and error to find the right COM port. It must have been a good omen.

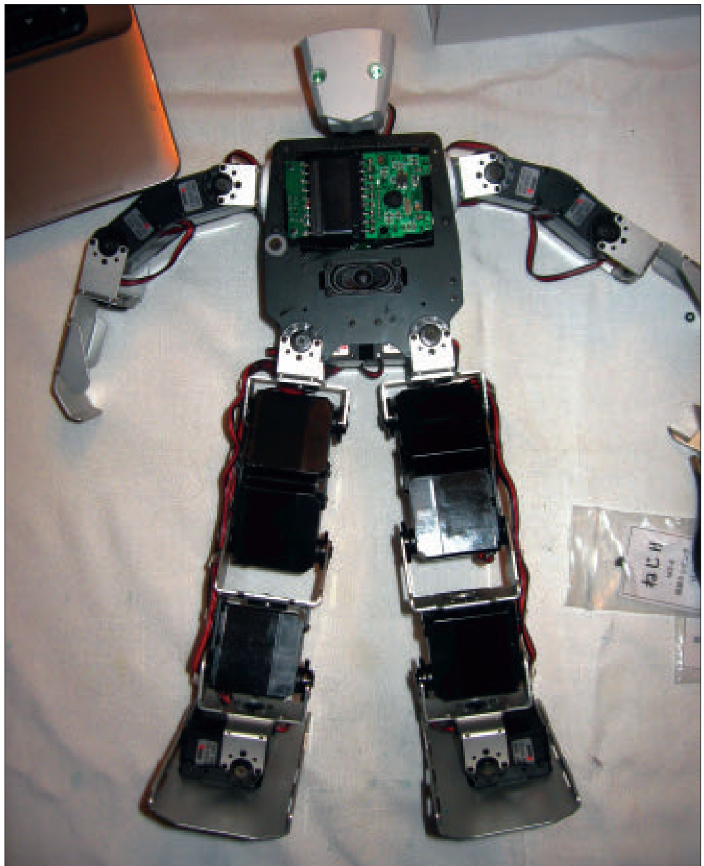
The software — RobovieMaker — comes with a pose editor and a motion editor. The basic pose editor features the familiar method of adjusting the position of each servo individually. Some details, however, raise the editor above the tedium that one might expect. Positions can be entered through individual clicks on up and down arrows, with each click corresponding to half of a degree. Positions can also be adjusted using sliders. The coolest part of the sliders is when the robot is connected and there is power to the servos. Another slider will track the actual position of the servo as it meets its new assignment.

The other half of the program is the far more sophisticated motion editor. The basic building units of the motion editor are pose blocks. Pose blocks basically contain the servo positions for a certain pose, and the blocks include a control for the speed of the transition between the previous and current poses. The default speed of 50 — already fairly quick

— is actually on the low end of the scale that ranges from 1 to 239. For fine adjustments and smooth motions, the user manual suggests using the duplication function to copy pose blocks. Another sage recommendation for stability is to use the reference pose like a voter for a Gilded Age political machine – early and often.

The accompanying CD comes with numerous preloaded motion files, ranging in category from dances to battle mode. Users can import existing motion files into a new motion for complex routines. The KT-X also comes with a speaker and numerous sound files. It can speak in Japanese and provide explosive sound effects to match its prowess in martial arts. Before programming our own poses, we thought to see what the default programs had to show us. But before that, the reference position of the robot had to be set. The KT-X default position is fairly close to the standing reference position, but users will need to make some small adjustments. We eventually had to go back and make the stance a little more stable, but the trial and error process goes pretty smoothly. After programming the reference pose, we loaded on some of the preprogrammed motions.

When left to its own devices, the KT-X will really show off its personality. It will do everything from say hello (in Japanese we think) to play the air guitar. A default program also allows the game controller to command the robot. Users can map their own commands to the game controller, and the numerous buttons on the pad allow the robot to have quite the extensive repertoire.



READY FOR SOME DANCE LESSONS.

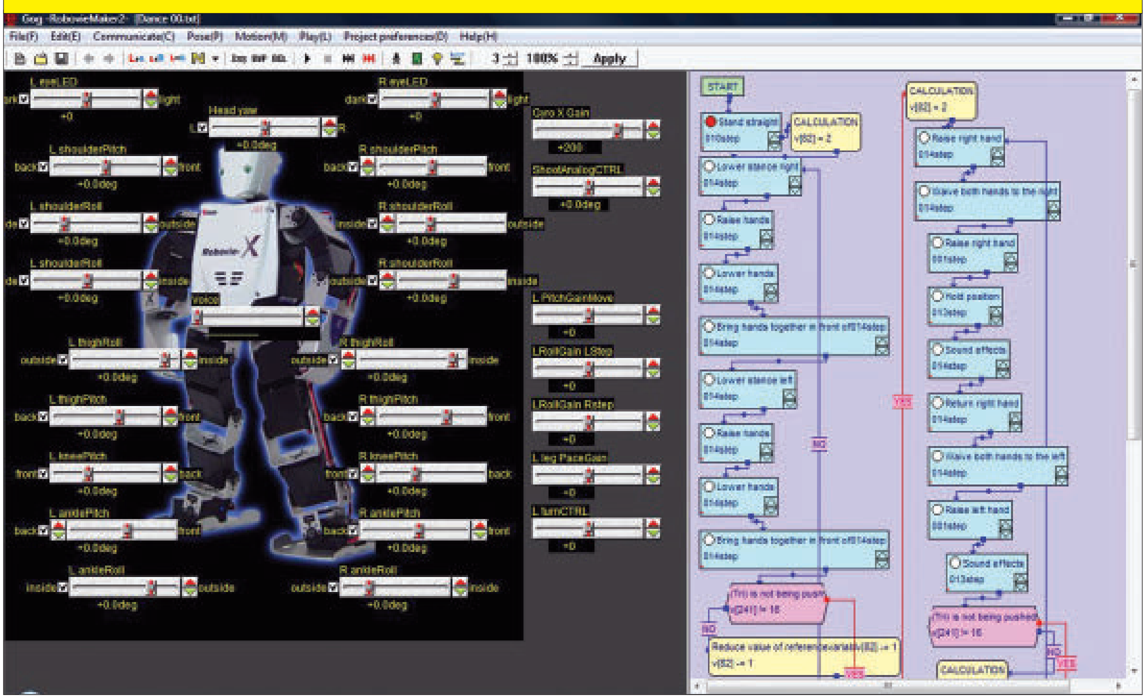
Dancing in the Dark

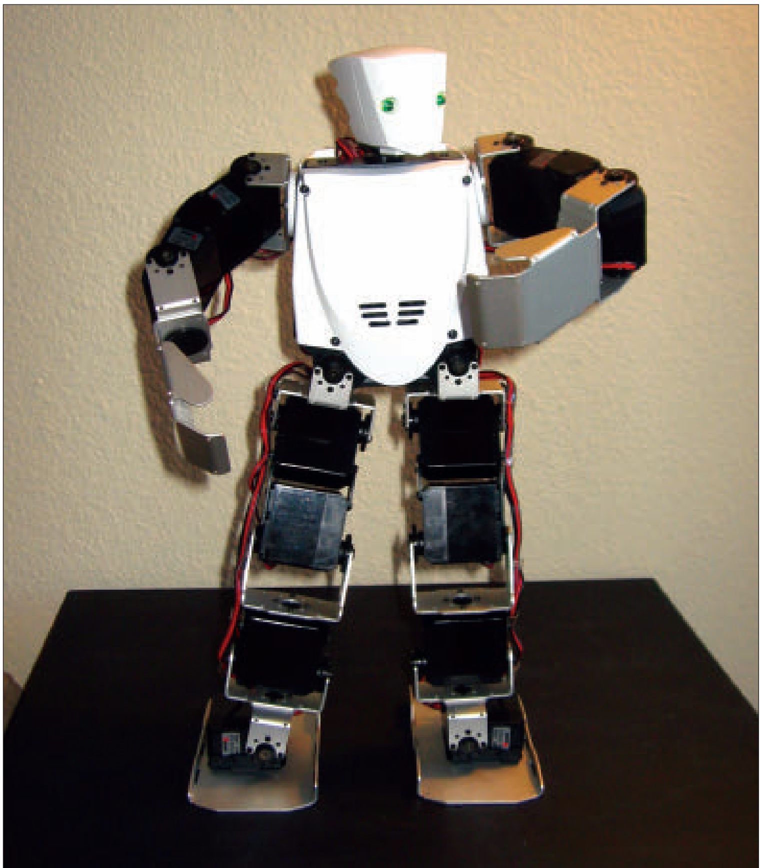
To hopefully capture the motion of our dancing robot, we planned to outfit it with an LED suit. This would also be a simple way to see if the KT-X was a promising platform for expandability.

The humble LED is a classic standby in circuit design, providing valuable assistance for everything from debugging to aesthetic value. And though the LED seems like a simple device, to pick the right one for your project you should consider several factors.

The first factor we examined was the viewing angle. Because we were aiming to create a visual spectacle, we wanted LEDs with large viewing angles. Many LEDS have small viewing angles,

THE ROBOVIEMAKER SOFTWARE.





Busting a move.

and can only be viewed by looking at them almost straight on. The LEDs we opted for had a 180 degree viewing angle which we thought would be plenty for a showman like the KT-X.

We also considered the power requirements. LEDs are usually happy with a few milliwatts, but it is still something to keep in mind. Because we wanted to use several LEDs in parallel, we didn't want them to be too power hungry. Most LEDs that popped up on Digi-Key had very modest power requirements, so this was not a hugely helpful way to narrow down the search.

The final consideration that helped us decide was the physical structure of the LED itself. We wanted the classic through-hole mount. It would be the easiest to solder and give us the simplest options for mounting to the robot. We did not want a surface-mount LED because the nearly nonexistent solder pads and miniscule scale would make them more trouble than they were worth. We settled on some through-hole mounted green LEDs that would nicely match the KT-X's glowing eyes. For several reasons that had more to do with timing than with the hackability of the KT-X, we also decided on the easy route of an external power source in the form of a nine-volt battery.

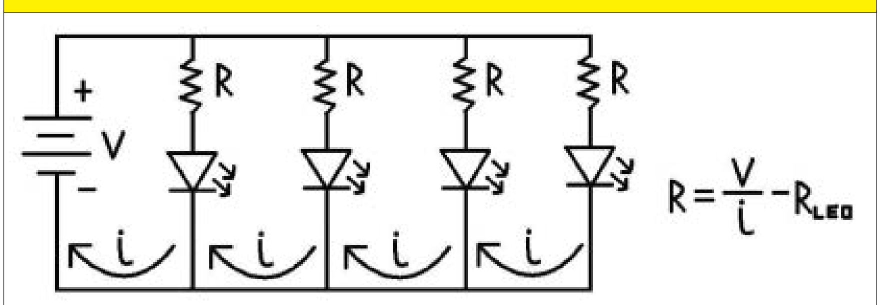
To properly implement the LED suit, we employed a classic current divider circuit. To ensure that each LED was the same brightness, we arranged them in parallel. Each LED was in series with a resistor to provide the proper input voltage. Calculating the optimum resistor value was a simple matter of recognizing that each loop contained the same current because the equivalent resistance of each loop was the same. The loop equivalent resistance was simply the sum of the LED resistance and the resistor value. The specs on Digi-Key provided the maximum current for the LEDs which was 20 mA. Assuming that 20 mA would be the maximum current in each loop, we were able to determine the resistance of the LEDs to be 110 ohms. The minimum resistor value was one simple calculation away after manipulating the classic formula of V = i*R. We calculated the minimum resistor value as 340 ohms. We ended up with 470 ohm resistors which might dim the LEDs slightly but definitely keep them from burning out.

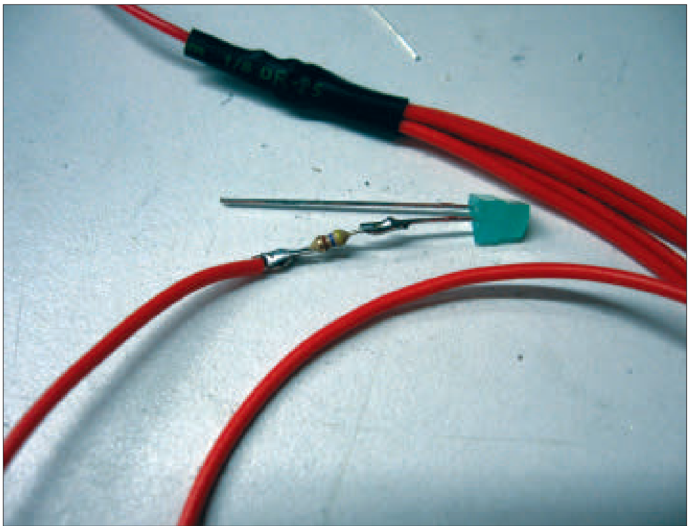
To connect the LEDs to the nine-volt battery, we simply soldered a bundle of wires to the leads off of a battery cap. Some heat shrink kept things looking nice, and the biggest challenge was keeping everything untangled. We took a cue from the KT-X design and used longer wires for the LEDs destined for the legs and shorter ones designated for the hands.

A current divider is admittedly a simple circuit, but it is always a good way to stay current on circuit design. And while this little circuit may seem a bit frivolous, it is reminiscent of far more sophisticated projects. Tracking joint movements with sensors is used in projects ranging from motion capture for animating movies and gait analysis for medical applications.

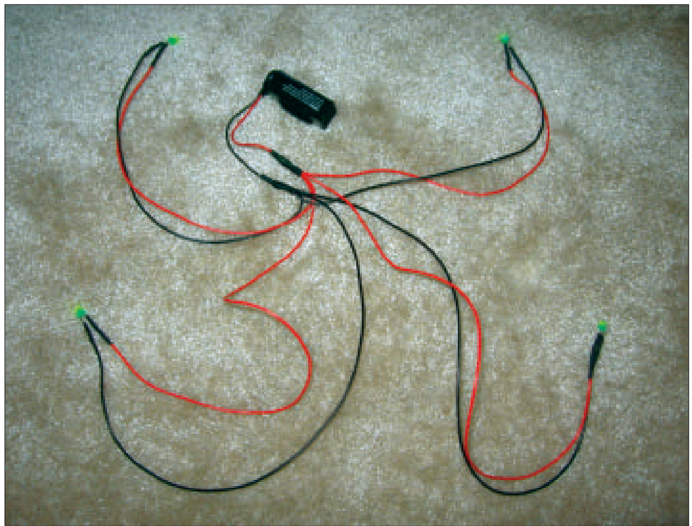
Attaching the LED suit to the KT-X really highlighted another great design feature bound to be of interest to tinkerers. While the robot does have an impressive 17 degrees of freedom, routing wires around the body is not nearly as risky business as it might sound. While the servos retain most of their 180 degree range of motion, the only trouble areas are the shoulders and hips. A little extra length of wire is all that's needed, so the addition of any other external sensors would likely not create an unmanageable tangle of red and black.

Schematic for a current divider circuit.





WIRING UP THE LEDS.



THE COMPLETED LED SUIT.

KumoTek Meets Discothèque

With the LEDs implemented, we were ready to program in some custom dance moves. The KT-X does come with some preprogrammed dance moves, complete with some groovy beats to dance to. We were aiming to recreate some classic dance moves, and to our knowledge the KT-X did not come with some preprogrammed disco funkiness. One of our concerns while dialing in our custom poses was power consumption; 17 servos can drain a battery fairly quickly. From past experience, we know that the bipedal humanoids tend to become a little erratic when on low batteries. The folks at KumoTek have once again anticipated such a problem. Firstly, when connected to the computer, you can choose whether or not to send power to the servos. When the robot is communicating with the computer, a window pops up showing the battery voltage. After a few hours of pose creation, the voltage was getting a bit low, but after recharging we were right back on track. To conserve power while modifying poses, users can also turn off power to individual servos.

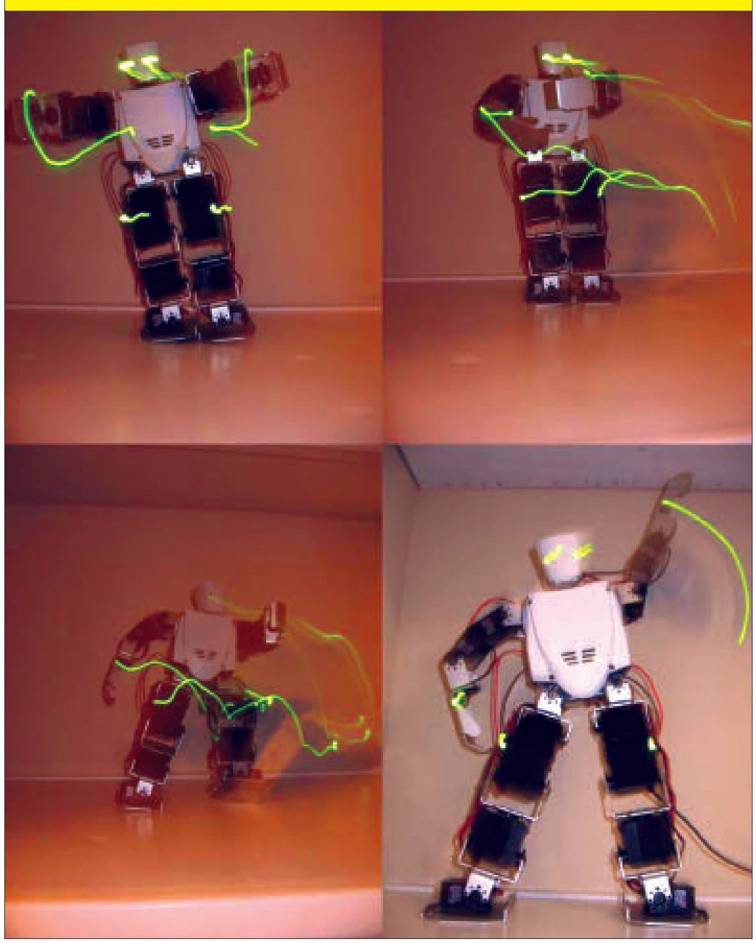
The CPU has a rotary dial with 10 settings, allowing users to download multiple programs to the robot. Motions can also be played while the robot is connected to the computer. Testing your own motions comes with a sense of accomplishment, especially when the LEDs trace the movements in the low lighting appropriate to a dance floor.

Gyromaniac

What may really set the KT-X apart from its competitors is the ability to add a gyroscope module. The module can be mounted to a free pin bank on the CPU board. The gyro allows the KT-X to maintain its balance — particularly during custom motion routines developed by the user. It can also detect when the robot has fallen down, allowing the KT-X to pick itself up autonomously.

To effectively implement the gyro, the folks at KumoTek have provided a sophisticated programming environment that goes far beyond creating poses by punching in the

KUMOTEK MEETS DISCOTHEQUE.



position of each servo individually. The motion editor has the ability to make conditional loops and to check the data from sensors like the gyro. The resulting flowcharts may look intimidating at first, but are easy to follow after becoming oriented with the programming manual.

The KT-X will run you about $1,290 which seems par for the course for the higher end servo humanoids. That price tag, however, includes a full warranty and great tech support. The folks from KumoTek are just a phone call away, ready with everything from a library of motion files and answers to all of your technical questions.

KumoTek offers several other expansion boards, as well. The selection includes an LED board (perhaps for the far more elegant version of our glam costume), a digital I/O board, and an analog I/O board. The expansion boards are a great way to add sensors to the robot for obstacle avoidance and more.

The Last Dance

Overall, we were supremely impressed with the KT-X. The folks at KumoTek have aimed to create a fun and functional bipedal humanoid that matches the quality of higher end models while remaining more affordable. We were definitely impressed with the quality of the kit, but what struck us most was the user friendliness. From little details like the clearly labeled and resealable bags to the constant reminders and tips in the construction manual, the KT-X certainly bucked the trend of tumultuous tedium that accompanies the construction of most bipedal humanoids. The RobovieMaker is also a great way to design your own poses and motions without too steep of a learning curve. The ability to connect and power the robot through the computer makes debugging a breeze, and even small details like the battery meter help users keep their robot charged and ready through the inevitable hours that they will want to spend with the Superbot. While many developers and users seem happy with the impressive accomplishment that a working bipedal humanoid undoubtedly is, the KT-X has gone further as a platform for expansion. The four recommended expansion boards add exciting capability, particularly with the gyro module. We're sure that intrepid hackers can add even more to the robot for comprehensive autonomous operation. **SV**

Recommended Websites

www.kumotek.com

Special Thanks To

Matthew Fisher

Then and NOW

ROBOT CARS

by Tom Carroll

The idea of robot cars has been around for many decades and, in fact, "automatic mobility" vehicles were envisioned from the very first automobile, though total autonomy as we know it today was not possible at the time. There have been many unique robotic developments that have been employed in various personal transportation devices. Science fiction movies and TV productions have long used robot cars in their sets, and most depictions are a bit off in their technology. The cartoon series, The Jetsons, showed George and family zipping about in some sort of flying car with nobody at the controls. i-Robot depicted Will Smith's policeman character zipping down streets and through tunnels under total automatic control in his Audi of the future (shown in ***Figure 1****). This concept car later became the more realistic Audi R8.*

For the average person, autonomous cars are just a vision of the future though BMW, VW, Mercedes, and many other luxury car brands now offer optional robotic features. The robot 'Johnny Cab' from the 1990 movie *Total Recall* is one autonomous personal vehicle that most people remember. **Figure 2** shows Arnold Schwarzenegger riding in back of the wayward robot cab. These thoughts of autonomy are not new. In the 1980s, Mercedes Benz built the world's first robot car, together with the team of Professor Ernst Dickmanns at Bundeswehr Universität in Munich, Germany.

DARPA is the True Winner of the Grand Challenges

Quite a few of us have followed the recent Defense Advanced Research Projects Agency Grand Challenges the past few years that have demonstrated the epitome of robotic technology as applied to cars. DARPA certainly had its own best interests in mind when it envisioned

FIGURE 1. The i-Robot Audi concept car.

FIGURE 2. The Johnny Cab from the film *Total Recall*.

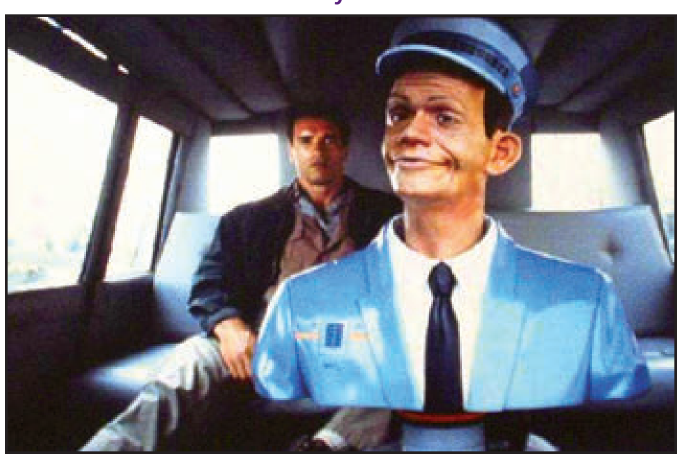


FIGURE 3. Autonomous DARPA Grand Challenge Ghostrider.

FIGURE 4. Stanford's Entry Stanley crosses the finish line.

an autonomous vehicle contest in the Mojave Desert of California with a $1 million prize. Starting from Barstow and ending 150 miles later near Las Vegas during March 8-13th, 2004, a field of 15 finalists, (including an autonomous motorcycle shown in **Figure 3**) began the race, after being weeded from a field of 100 applicants. Most crashed seconds after starting; eight barely made it a single mile; and the two best made it eight miles. It was a start, and a lot was learned from this seemingly group of 'losers.' I remember thinking back then that I knew the race would be hard to complete, but afterwards I wondered if anyone could complete it with 2004 technology.

Just a year and a half later on Sunday, October 8, 2005, a modified Volkswagen Touareg SUV/crossover named Stanley (shown in **Figure 4**) from the Stanford (University) Racing Team beat 23 other robotic cars on 132 miles of hot and unforgiving Nevada desert to claim a $2 million prize. Stanley used seven Pentium-class computers, along with virtually the same sensor suites used by the other entrants such as GPS, Lidar, and video pattern

FIGURE 5. Carnegie Mellon's Tartan racing team won the Urban Challenge.

recognition, to not only know the route to take (GPS) but to see where it was and where it should not be. The team formed by Red Whittaker, Carnegie Mellon robotics professor and one of the world's greatest driving forces for robotics, came in second and third with their Sandstorm and H1ghlander vehicles.

By 2007 — with robot cars becoming more sophisticated — DARPA held an Urban Challenge on a mock 'urban street array' in California. On Saturday, November 3, 2007, the Tartan Racing Team (Red's team from Carnegie Melon) entry named Boss bested Stanford's Stanley Jr., Virginia Tech's entry, and others in the 60 mile course. The winning Carnegie Mellon team is shown in front of Boss in **Figure 5**. This event (not a race) was not without its problems as Georgia Tech's Porsche vehicle ran into a barrier and mangled its sensor bar. Other vehicles unexpectedly died. Stanford and VT finished second and third respectively. Participating vehicles had to cross intersections with competitors, pass others, deal with professional drivers on the course, and obey basic traffic rules.

The Rules were simple, so to speak. Here's a list of them from Carnegie Mellon's site:

- Follow rules of the road.
- Detect and track other vehicles at long ranges.
- Find a spot and park in a parking lot.
- Obey intersection precedence rules.
- Follow vehicles at a safe distance.
- React to dynamic conditions like blocked roads or broken-down vehicles.

The US military was given a mandate from Congress to have 30% of all military vehicles unmanned by 2015. The DARPA Grand Challenge and Urban Challenge were part of our military's method in achieving this goal. Sponsor a series of contests and have the best and brightest students, engineers, and scientists from top universities duel for some pretty nice prizes. That's a lot cheaper than doling out a

FIGURE 6. The Lexus LS460 parking itself.

few hundred million dollar study contracts to aerospace and industrial firms. Hopefully, a few military contractors can take the data learned from these contests and build some nice, million dollar robot military vehicles. In the same manner, automobile manufacturers can take the same data and build $50,000 robot cars for the average driver.

Robot Cars Available Today

There are already some pretty smart features available on today's vehicles. Car manufacturers are not even close to applying totally autonomous sensor/computer systems that were used in the DARPA contests, but the major manufacturers are closing in. Automatic parking is one recent feature that still amazes people as they watch the top-end Lexus park itself. The LS460 Advanced Parking Guidance System has been available for about five years and assists drivers in one of the more difficult driving scenarios: parallel parking.

Obviously, one has to pick a potential parking space with enough room for your car because the system will try to fit into any space you select. When the proper space is selected, all the driver has to do is pull up beside the 'front' car. An active sonar transducer on the front bumper measures the distance from the front car, as well as the length of the potential parking space. The car is put into reverse and a rear-mounted camera presents a wide-angle view of the rear car area. You then push the 'Parallel-Park' icon on the dash-mounted navigation system touch screen. When a green icon on the screen is centered on the space, you press 'OK,' take your hand off the wheel, and the car automatically steers itself into the space with only your foot on the brake to stop it at the end. There are other conditions to consider, but, for a person who usually has to try multiple times to park, this system is a cool way to go. **Figure 6** shows the Lexus parking itself to the amazement of the observing crowd.

Other car companies have jumped onto the bandwagon with their own versions of automatic parallel parking vehicles. The Toyota Prius would never be considered a luxury car (or a particularly stylish vehicle) but the top of the line Prius T-Spirit has an automatic parking system similar to the Lexus, as they are both made by Toyota. Cadillac has recently announced the development of a hydrogen-powered, driverless car nicknamed the 'Boss.' That famous center of robotics technology — Carnegie Mellon University — has teamed up with General Motors to make this car happen. Car companies are anxious to bring out the latest in technology, as well as offer drivers easier ways to drive long and boring distances.

FIGURE 7. Four Robot Vans start out from Milan.

Cars have long had speed controls so a driver can take his or her foot off the gas pedal and stay at a desired speed. Ultrasonic sensors in the front and back bumpers alert a driver to other cars or obstacles in front or behind the car. RF (radar) sensors detect cars in excess of a hundred feet in front of vehicles and adjust speed and closing distance accordingly to avoid accidents. Add in real time GPS data and moving maps upon a screen, GPS enhanced voice guidance to a destination of your choice, self-parking, weather, and traffic updates, and today's top-end cars are truly marching towards total autonomy. Add-ons that were once considered cool gadgets are now must-haves for many car buyers.

Robot Vans Travel from Italy to China

As you're reading this, two driverless 'robot' vans should be completing a trip from Milan, Italy to Shanghai, China — an 8,000 mile journey. Developed by VisiLab (a division of the University of Parma, Italy that specializes in computer vision systems and AI), this test will pave the way for the development of autonomous delivery vehicles. The four small vans shown in **Figure 7** have seven cameras and four laser scanners each to develop a computer image of

FIGURE 8. Interior of the robot van that can accommodate two drivers.

FIGURE 9. Robot van and solar panel.

the surrounding road, the road's lanes, and the location of other vehicles and pedestrians. The computers in each vehicle combine this sensory data in conjunction with GPS data, waypoint following, and other navigation devices to allow the vans to travel autonomously. Each van has two human passengers to take over in case of emergencies. I've followed the blog and a few web videos, and they were only just reaching Kazakhstan in early September with quite a way left to go.

The purpose of this $2.3 million project was not to have cars traverse smooth city streets and freeways with tons of map and street data available, but to travel on rough, back-country roads at intercontinental distances with little or no mapped data. The Piaggio Porter Electric vans are 'green,' solar-powered test beds with a lead vehicle determining the way and data sent to the following van. **Figure 8** shows the interior of one of the vans and **Figure 9** shows a close-up of the van and its photovoltaic roof-mounted panel. Two support vehicles contain generators to charge batteries out in the 'middle of nowhere,' as well as tools and spare parts. Humans can take over when road decisions have to be made or unexpected situations occur. Other trucks will follow the team with supplies, accommodations, storage, and a machine shop. Traveling at only 37 mph at four hours a day due to the need for recharging, the European Research Council and the University of Parma sponsoring the trip will gather over 100 terabytes of data that can later be used for potential commercialization purposes.

Robot Cars Climb Mountains

Sliding into a parking space or even driving a rough road that is flat is certainly a bit easier than maneuvering around 156 tight turns over 12 plus miles on a dusty gravel road, while traveling at speeds of up to 90 mph, all under computer control. We're talking going straight up Pikes Peak in Colorado — a mountain over 14,000 feet high. The part paved, gravel and dirt road that rises 4,721 feet in altitude at an average 7% grade, tested professional drivers this past June at the prestigious Pikes Peak International Hill

FIGURE 10. The Audi TTS 'Shelly' will climb Pikes Peak.

FIGURE 11. Photo by Jim Merithew of Wired.com shows Shelly's computers.

Climb. In September of this year, multiple autonomous vehicles will have attempted this difficult task. One of the cars is an Audi TTS nicknamed 'Shelly' (shown in **Figure 10**) that was specially developed for the race by Stanford University in cooperation with Volkswagen's Electronics Research Laboratory (ERL) based in Silicon Valley. **Figure 11** shows a small amount of the computing power of Shelly.

ERL is no stranger to designing and building robot cars as it was the group that took the re-worked 2005 VW Stanley I mentioned earlier to win DARPA's 132 mile robot car race through the Mojave. ERL (and Stanford's AI Lab) later built Junior based on a VW Passat that was designed to autonomously maneuver within slow city traffic — a vehicle more adaptable for today's average consumer driver. Marcial Hernandez, a project manager at ERL, says that VW's Junior "was about getting you to Grandma's house" — slow, meticulous, safe. In contrast to the much slower Junior, Shelly will scream up the winding road at breakneck speeds and probably give Grandma a heart attack, and certainly her children riding in the car. These various stepping stones to more advanced automotive technology around the world will continue until the perfect robot car for the average driver is available in dealer showrooms. Sure, it's marketing, but it is also a way to stay in business in the competitive automotive industry in these difficult times.

Why Have a Robot Car?

That might seem to be a thoughtless question in a robotics magazine in these days of great technological advances, but many people have truly wondered just why we need all this automation. The proponents of driverless cars speak of safety, the convenience of calling a robo-taxi from your cell phone, less congestion on roads, and robotized delivery. They naively speak of no more DUIs, 41,000 highway deaths per year would no longer happen, and kids could be safely delivered to their schools. The detractors also speak of safety or the lack thereof, extreme liability exposure, the high costs of the required infrastructure, and reliable vehicles.

Getting a little deeper, non-technical people have questioned "What if a fuse blows or a computer fails when my driverless car is barreling down a highway of the future?" "Why give the 'passengers' of a driverless car a false sense of security when something can so easily go wrong?" "Can these robot cars work on our regular highways?" Technical people ask "What will power these cars as present battery technology — even today's Li-Ion batteries — seem woefully inefficient." "The many sensors and multiple computer systems required for today's DARPA and other contests make the successful development of these cars a very high hurdle."

Well, in response to all of these very valid questions, nobody has ever said that the development and implementation of any technology would be easy. Cell phone systems were first laughed at by intelligent people as impossible. "Having thousands of personal transceivers interconnecting with a myriad of cell towers would be a logistical nightmare." Well, the first cell phones cost thousands of dollars, were as big as a brick, and only the wealthy could afford the high cost of using them. Today's tiny cell phones are literally given away, have so many functions that few people use them all, plus cell towers are everywhere for the billions of people who use cell phones worldwide. If people want driverless cars that are safe and save them time and money, they will come as soon as the public demands them.

What is Needed to Make Autonomous Cars a Reality?

Some have said that the first real step in implementing driverless cars is to have them operate on a separate roadway away from human-driven cars. The Personal Rapid

FIGURE 12. Driverless pod for use at London's Heathrow Airport.

FIGURE 13. Heathrow pod roadway.

Transit System due to be installed at London's Heathrow Airport this year can carry four adults and their luggage. When I was returning from Copenhagen last year, I saw a mockup of one of these 'pod cars' similar to **Figure 12**. They were also on exhibit at London's Science Museum. Designed only for short hauls, they will be the ideal way to go from a parking lot to a terminal. **Figure 13** shows the dedicated roadway for these pods.

Airports such as Seattle, Newark, and many others have dedicated roadways for terminal trains, but ones designed for robot cars is a first. Automated distance markers, autonomous acceleration and deceleration, obstacle (other pods) detection, and destination information will make these short trips easy for travelers. Longer distance dedicated roadways for inter-city personal transportation is the next step with the ability to use standard roads the next goal.

Intelligent systems integrated into our nation's roads are the keys to keeping autonomous car traffic safe. The actual roads will keep track of all the cars, their speeds, their entry and exit points, determine following distances, and report any problems to all cars on the roadway. The individual cars will report any problems to the system so that the following cars can adjust accordingly. Cars, themselves, will be able to talk with each other on the new FCC 5.9 GHz band that the National Highway Transportation Safety Administration is considering in conjunction with the Intelligent Transportation System. Auto RFID toll systems such as Fast Trak in California and other parts of the country will be enhanced for multiple data input and output. Rural roads and highways can also be enhanced to offer vehicles in sparse traffic safety, weather, and road information, and instant emergency communications such as GM's OnStar system that alerts emergency personnel of an accident.

Final Thoughts

Despite the freedom-loving American public's desire to do and go anywhere they might want with no restrictions, more and more people and groups are looking towards autonomy in their automobiles. Four billion hours of wasted time and almost three billion gallons of gas due to traffic delays make autonomous robot cars sound like the way to go. Scientists and students in universities around the country and world are pushing closer to a totally autonomous robot car.

I've just touched upon the reason "why" to build a robot car in this article. The "when" is up to you. **SV**

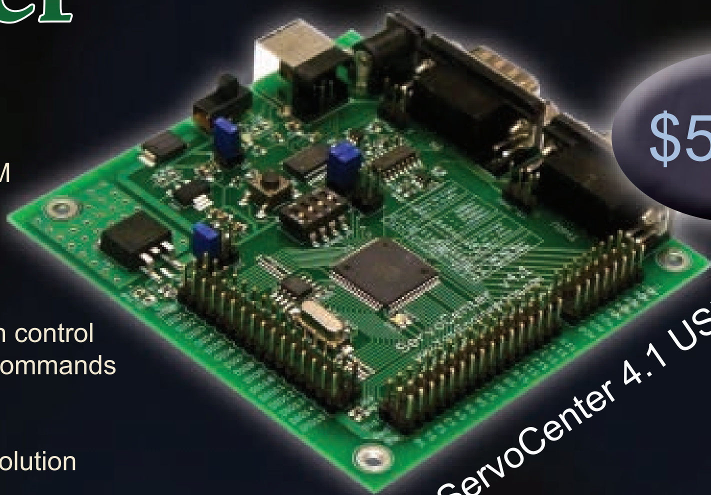
Tom Carroll can be reached at TWCarroll@aol.com.

THE OWNERSHIP, MANAGEMENT, AND CIRCULATION STATEMENT OF SERVO MAGAZINE, Publication Number: 1546-0592 is published monthly. Subscription price is $24.95. 7. The complete mailing address of known office of Publication is T&L Publications, Inc., 430 Princeland Ct., Corona,Riverside County, CA 92879-1300. Contact Person: Tracy Kerley. Telephone: (951) 371-8497. 8. Complete Mailing address of Headquarters or General Business Office of Publisher is T&L Publications, Inc., 430 Princeland Ct, Corona, CA 92879. 9. The names and addresses of the Publisher, and Associate Publisher are: Publisher, Larry Lemieux, 430 Princeland Ct., Corona, CA. 92879; Associate Publisher, Robin Lemieux, 430 Princeland Ct., Corona, CA 92879. 10. The names and addresses of stockholders holding one percent or more of the total amount of stock are: John Lemieux, 430 Princeland Ct., Corona, CA 92879; Larry Lemieux, 430 Princeland Ct., Corona, CA 92879; Audrey Lemieux, 430 Princeland Ct., Corona, CA 92879. 11. Known Bondholders, Morgagees, and other security holders: None. 12. Tax Status: Has not changed during preceding 12 months. 13. Publication Title: SERVO Magazine 14. Issue Date for Circulation Data: October 2009-September 2010. 15. The average number of copies of each issue during the proceeding twelve months is: A) Total number of copies printed (net press run); 11,825 B) Paid/Requested Circulation (1) Mailed Outside County subscriptions: 5,043 (2) Mailed In-County subscriptions: 0 (3) Paid Distribution Outside the Mail including Sales through dealers and carriers, street vendor, and counter sales and other paid distribution outside USPS: 2,423 (4) Paid Distribution by other classes of mail through the USPS: 0; C) Total Paid Distribution: 7,466; D) Free or Nominal Rate Distribution by mail and outside the mail (1) Free or Nominal Rate Outside-County Copies: 0 (2) Free or Nominal Rate In-County Copies: 0 (3) Free or Nominal Rate Copies Mailed at other classes through the USPS: 0 (4) Free or Nominal Rate Distribution Outside the mail: 350; E) Total Free or Nominal Rate Distribution: 350; F) Total Distribution: 7,816; G) Copies not distributed: 4,009; H) Total: 11,825; Percent paid circulation: 95.52%. Actual number of copies of the single issue published nearest the filing date is September 2010; A) Total number of copies printed (net press run) 13,223; B) Paid/Requested Circulation (1) Mailed Outside County subscriptions: 4,670 (2) Mailed In-County subscriptions: 0 (3) Paid Distribution Outside the Mail including Sales through dealers and carriers, street vendor, and counter sales and other paid distribution outside USPS: 2,941 (4) Paid Distribution by other classes of mail through the USPS: 0; C) Total Paid Distribution: 7,611; D) Free or Nominal Rate Distribution by mail and outside the mail (1) Free or Nominal Rate Outside-County Copies: 0 (2) Free or Nominal Rate In-County Copies: 0 (3) Free or Nominal Rate Copies Mailed at other classes through the USPS: 0 (4) Free or Nominal Rate Distribution Outside the mail: 550; E) Total Free or Nominal Rate Distribution: 550; F) Total Distribution: 8,161; G) Copies not distributed: 5,062; H) Total: 13,223; Percent paid circulation: 93.26%. I certify that these statements are correct and complete. Larry Lemieux, Publisher - 9/30/10.

Advertiser Index

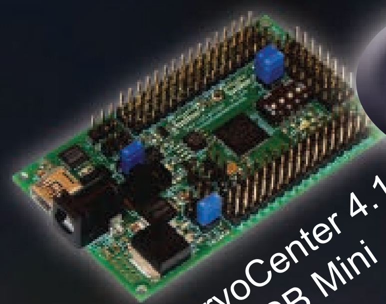
The Future of Servo Control is Calling...

ServoCenter™



Features

- USB, RS232, and TTL serial support
- 16 servos, 16 digital I/O, 8 analog inputs
- Built-in SC-BASIC Sequencer with EEPROM
- Sequencer allows stand-alone operation
- 64 scene presets stored in EEPROM
- Presets instantly loaded or cross-faded
- Built-in configurable smoothing algorithm
- Independent, simultaneous speed & position control
- Scaled, percentage, and group movement commands
- Timed movement commands
- Max, min, & startup position settings
- Ultra-precise 0.05425µS jitter-free pulse resolution

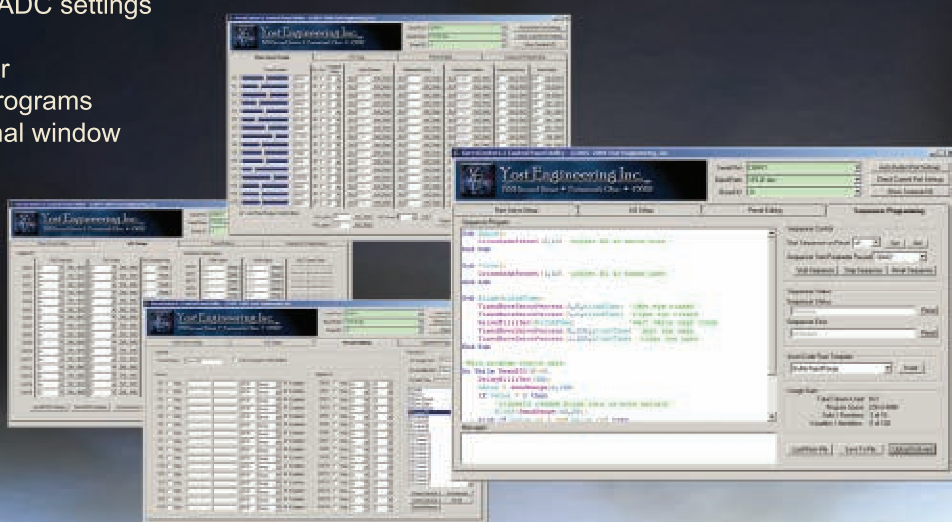


Flexibility

- User upgradeable firmware
- Upload your own firmware with bootloader
- Watchdog timer for failsafe operation
- Over-current, over-temperature, polarity protection
- Internal regulator, external power, or battery power options
- Supports 4.8/6.0V regulated servo supply voltages at 5 Amps
- Each digital I/O and analog input has supply pins
- Direct serial, activeX control, or Win32 DLL communication
- Programming examples in 10+ languages
- Windows, Linux, Mac OSX compatible

Free Control Panel Software

- Easy editing and configuration of servo settings
- Configure and set up digital I/O & ADC settings
- Edit scene presets
- Program the SC-BASIC sequencer
- Upload and run your SC-BASIC programs
- Debug & communicate with terminal window



www.Servo-Center.com

Yost Engineering Inc.


NO MATTER WHAT THE IDEA
YOUR PCB PROTOTYPES SHOULD BE THE EASY PART
QUOTE & ORDER PCBS ONLINE AT WWW.SUNSTONE.COM OR CALL 1-800-228-8198


ValueProto™
PCBexpress®
Full Feature